

Universidade Federal do Espírito Santo
Centro Tecnológico
Programa de Pós-Graduação em Engenharia Elétrica

Leonardo de Assis Silva

Aproximação Planar por Partes Para Reconstrução 3D Densa

Vitória

2016

Leonardo de Assis Silva

Aproximação Planar por Partes Para Reconstrução 3D Densa

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Universidade Federal do Espírito Santo

Centro Tecnológico

Programa de Pós-Graduação em Engenharia Elétrica

Orientadora: Profa. Dra. Raquel Frizera Vassallo

Vitória

2016

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Setorial Tecnológica,
Universidade Federal do Espírito Santo, ES, Brasil)

S586a Silva, Leonardo de Assis, 1992-
Aproximação planar por partes para reconstrução 3D densa
/ Leonardo de Assis Silva. – 2016.
122 f. : il.

Orientador: Raquel Frizera Vassallo.
Dissertação (Mestrado em Engenharia Elétrica) –
Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Reconstrução de imagens. 2. Imagem tridimensional.
3. Regiões homogêneas. 4. Triangulação. I. Vassallo, Paula
Frizera. II. Universidade Federal do Espírito Santo. Centro
Tecnológico. III. Título.

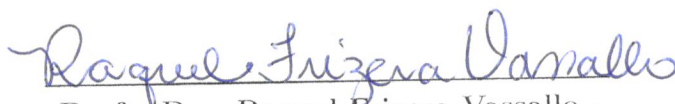
CDU: 621.3

Leonardo de Assis Silva

Aproximação Planar por Partes Para Reconstrução 3D Densa

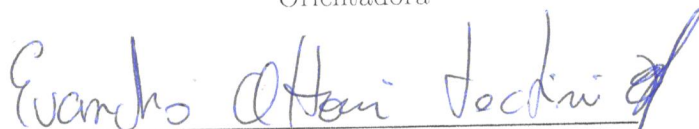
Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Trabalho aprovado. Vitória, 16 de dezembro de 2016:



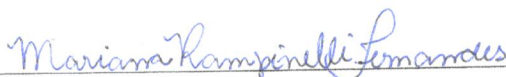
Profa. Dra. Raquel Frizera Vassallo -
UFES

Orientadora



Prof. Dr. Evandro Ottoni Teatini
Salles - UFES

Membro Interno



Profa. Dra. Mariana Rampinelli
Fernandes - IFES

Membro Externo

Vitória

2016

Agradecimentos

Aos meus pais, Valdir e Luciene, e ao meu irmão, Lucas. Família não se escolhe, e por isso eu tive muita sorte. Obrigado por sempre me apoiarem e por todo amor e carinho que me deram nesses 25 anos. Vocês sempre lutaram para que eu crescesse tanto como profissional quanto ser humano. Sei que vou poder contar sempre com vocês.

À Carolina, por ser minha companheira por esses últimos 6 anos. Seu carinho e amor sempre me acalmaram, aquecendo meu coração. Obrigado por me mandar mensagens meigas e engraçadas quando vou dormir. Obrigado pelos desenhos e presentes que me deu só pra me ver feliz c:.

Ao Thiago, ao Ícaro e ao Tadeu por acharem que são donos de parte do meu carro e me fazerem rir disso. Obrigado por jogarem as minhas histórias de RPG “excelentes” e por estarem sempre comigo (Tadeu está internacional agora, mas vai continuar sempre nas histórias salvando os reinos). Vocês são ótimos amigos.

À Prof^a. Raquel, por aceitar um “Raquel, passei no mestrado e você vai ser minha orientadora!” e me orientar por 2 anos. Agradeço também por você não ser só uma orientadora e sim uma grande amiga, e por fazer do laboratório uma segunda casa para mim e para todo mundo do Viros. Lá eu aprendi a diversão que é dar aula e pesquisar em equipe. Minha mainha já aceitou que você é uma segunda mãe, capitã dos pinguins.

À Thaís, por aguentar os infinitos pedidos de ajuda, consertar meu texto, escrever meu texto, fazer algumas “poucas” figuras e aceitar ser minha *coach*. Parece que não dá pra dar meio título de mestrado, mas esse agradecimento, a felicidade e eu ter aprendido a mexer no Inkscape já é recompensador né :D?! Você é uma grande amiga.

À Netalianne, por ser uma grande pequena-amiga. Sempre pude contar com você na graduação, no mestrado e até pra estudar pra concurso. Estarei contando com você para darmos aula. Obrigado pela grande amizade, Netal.

Ao Prof. Evandro, por aceitar me orientar inicialmente, me acolher no seu laboratório e me tratar como um membro de lá. Agradeço muito, professor.

Aos meus companheiros de empresa Felipe e Ricardo. Vocês me acolheram no laboratório e hoje somos uma equipe de pinguinzões. A empresa atualmente é formada por 1 mestre, 1 engenheiro e 1 técnico, que na verdade possuem os cargos respectivos na empresa de “quebrador oficial de qualquer coisa”, “faz-tudo com acabamento grosseiro” e “faz-tudo com acabamento fino”. É uma honra trabalhar com vocês, rapazes o>. Obrigado por tudo.

À Junia e ao Rodolfo, por escreverem meu texto, meus códigos e ainda me chamarem

de amigo e inseto, respectivamente! Não era exploração, eu só gosto de trabalhar com vocês e com a galera do laboratório! Obrigado por tudo, galera.

Ao Demuth, ao Daniel, à Thaís, ao Fernando, ao Thales, ao Legolas e ao Matheus, grandes amigos, por sempre pararem o que estavam fazendo para tentar me ajudar mesmo quando também estavam correndo contra o tempo.

É muito bom trabalhar com vocês de ambos laboratórios. Passar o tempo com vocês sempre foi revigorante. Agradeço à todos vocês também por depois de vários seminários e conversas malucas terem gravado que existe uma “linha epipolar” e que isso era importante pra mim.

Em suma, à todos meus amigos e companheiros de laboratório.

Aos Profs. Edinho, Diego, Raquel, Mariana, Alexandre, Flávio, Fabrício e Evandro por torcerem por mim na minha carreira acadêmica. Eu me espelho em vocês e aprendi muito com todos. Obrigado.

Ao Christiano, pela ajuda neste trabalho. Obrigado por todo tempo, toda atenção e paciência.

À Deus que me fez chegar onde cheguei e sempre zelou tanto por mim quanto pela minha família e por todos que amo. Muito obrigado.

À CAPES pela bolsa concedida e ao PPGEE, pelo mestrado oferecido. Aos professores, pelas disciplinas ofertadas.

Amigos e companheira a gente escolhe, e eu aprendi com minha família a escolher bem. Tenho muita sorte de ter encontrado todos vocês para fazer parte da minha grande família de pinguins. Obrigado à todos.

Resumo

Com o aumento da capacidade de processamento dos dispositivos, imagens podem ser usadas para analisar cenas e extrair informações tridimensionais dos pixels em um espaço de tempo bem menor do que se conseguia fazer há alguns anos atrás. Esse processo de recuperação da informação tridimensional do meio recebe o nome de reconstrução 3D. Estimar com precisão a profundidade de regiões homogêneas de uma imagem ainda é um desafio na área. Neste trabalho, foram propostas duas metodologias de reconstrução 3D densa aproximando-se regiões homogêneas por planos e por uma superfície obtida a partir da triangulação de Delaunay. Essa abordagem tem como objetivo apresentar um bom compromisso entre precisão e velocidade. O trabalho está focado apenas na etapa de reconstrução, sendo a estimativa de parâmetros intrínsecos considerada obtida por calibração e os parâmetros extrínsecos resultantes de um processo de *tracking* da câmera. Para a estimação da reconstrução, é utilizado um conjunto de 10 imagens que possuem certa sobreposição com uma imagem de referência. O correspondente de cada pixel da imagem de referência é procurado nas 10 imagens resultando em uma estimativa esparsa após uma etapa de fusão de informações e filtragem de possíveis *outliers*. É possível diminuir o tempo de processamento do algoritmo, utilizando-se uma pirâmide gaussiana de multiresolução. As regiões homogêneas são identificadas com a técnica SRM, tendo sua profundidade estimada a partir da nuvem de pontos esparsa reconstruída a priori utilizando uma das duas metodologias propostas neste trabalho: aproximação por planos ou por uma superfície obtida a partir da triangulação de Delaunay. Os resultados obtidos foram satisfatórios segundo os critérios de precisão, *recall* e tempo de processamento. Comparando-se com o DTAM, uma técnica de tempo real, os resultados obtiveram melhor precisão, principalmente em regiões homogêneas. O maior tempo de processamento é devido ao *hardware* de menor capacidade quando comparado ao caso do DTAM. Além disso, o código implementado não está otimizado. Desta forma, é possível se obter velocidades maiores de reconstrução com o método proposto. De acordo com as avaliações feitas neste trabalho, os resultados obtidos podem ser considerados promissores.

Palavras-chave: Reconstrução 3D densa. SRM. Regiões homogêneas. Aproximação planar por partes. Triangulação de Delaunay

Abstract

Considering the increase in device processing capabilities, images can be used to analyze scenes and extract three-dimensional information from pixels. The process of retrieving the three-dimensional information from the environment is called 3D reconstruction. Estimating precisely the depth of homogeneous regions in an image is still a challenge in computer vision. In this work, two dense 3D reconstruction methods were proposed, approximating homogeneous regions by planes and by a surface obtained from Delaunay triangulation. Our approach aims to achieve a good trade-off between precision and processing time. It is worth to mention that this work is focused only on the reconstruction stage. The intrinsic parameters are considered to be obtained by calibration, while the extrinsic parameters are estimated from a camera tracking process. The reconstruction estimation process needs a set of 10 images that presents some overlap with a reference image. The matching for every pixel in the reference image is searched in all the 10 images, resulting in a sparse estimation after a fusion and filtering stage. It is possible to decrease the algorithm processing time by using a gaussian multiresolution pyramid. The homogeneous regions are identified with SRM technique and their depth estimation is obtained from the sparse points cloud reconstructed a priori. This is done using one of the methods propoused: the planar aproximation or the surface obtained through Delaunay triangulation. The results were satisfactory according to the criteria of precision, recall and processing time. Comparing with DTAM, a real-time reconstruction technique, the results achieved better accuracy, especially in homogeneous regions. The longer processing time is due to the lower capacity hardware when compared to the hardware used for the DTAM experiments. Additionally, the implemented code is not optimized. In this way, it is possible to obtain better reconstruction results with the proposed method. Considering the evaluations carried out in this work, the obtained results may be considered promising.

Keywords: Dense 3D reconstruction. SRM. Piecewise planar aproximation. Delaunay triangulation

Lista de ilustrações

Figura 1 – Quantidade de trabalhos que abordam o tema de reconstrução 3D (cinza escuro) dentro da área de visão computacional (cinza claro). Esse quantia também está apresentada em percentual da área de visão computacional.	22
Figura 2 – (a) Reconstrução 3D do método DTAM. (b) Reconstrução 3D da técnica TMVS.	23
Figura 3 – Reconstrução 3D da técnica PMVS.	24
Figura 4 – Reconstrução 3D do sistema proposto por Alcantarilla, Beall e Dellaert (2013).	25
Figura 5 – Reconstrução 3D do sistema proposto por Concha e Civera (2015). (a) Reconstrução esparsa. (b) Aproximação planar de regiões homogêneas. (c) Resultado da junção de (a) e (b).	26
Figura 6 – Reconstrução 3D do sistema proposto por Romanoni e Matteucci (2015). Reconstrução utilizando triangulação de Delaunay com (b) e sem (a) textura.	27
Figura 7 – Representação do registro de uma cena por uma câmera no modelo de câmera <i>pinhole</i>	30
Figura 8 – O sistema de coordenadas da câmera (a) pode ser representado também como (b) onde o eixo Z intercepta o plano da imagem no ponto principal \mathbf{c}_o . Um ponto tridimensional \mathbf{X} é projetado no plano da imagem no pixel \mathbf{u} . (c) apresenta essa projeção pela vista do plano YZ	31
Figura 9 – Translação da referência no plano da imagem do ponto principal \mathbf{c}_o para o canto superior esquerdo.	32
Figura 10 – Transformação (R_{wc}, t_{wc}) do sistema de coordenadas da câmera S_c para o global S_w	34
Figura 11 – A foto que brinca com o conceito de perspectiva (a) pode ser também conferida de outro ângulo (b) que mostra a localização da flor e da menina na sala onde a foto foi tirada. Em (b) pode-se notar por meio de linhas que a projeção da cena na matriz de sensores da câmera terá a flor com mesmo tamanho que a saia da menina.	35
Figura 12 – (a) Projeção do ponto tridimensional \mathbf{X}_1 no plano da imagem da câmera \mathbf{O}_1 . (b) Os pontos $\mathbf{X}'_1, \mathbf{X}''_1$ e \mathbf{X}'''_1 sobre a linha ℓ_1 também possuem mesma projeção que \mathbf{X}_1 em \mathbf{u}_1	36
Figura 13 – Triangulação do ponto \mathbf{X}_1 com a adição de uma segunda imagem. (R_{21}, t_{21}) representa a transformação de \mathbf{X}_1 no sistema de coordenadas da câmera 1 para \mathbf{X}_2 no sistema da câmera 2.	37

Figura 14 – (a) pixels correspondentes a $\hat{\mathbf{u}}_1$ na imagem 2 para diferentes valores de z_1 . (b) A projeção da linha de projeção ℓ_1 no plano da imagem da câmera 2 recebe o nome de linha epipolar.	38
Figura 15 – O ponto \mathbf{e}_1 , chamado de epipolo , é a projeção do centro óptico da câmera 2 (\mathbf{O}_2) no plano da imagem da câmera 1. O epipolo \mathbf{e}_2 é encontrado da mesma maneira. O plano π é formado pelos pontos \mathbf{O}_1 , \mathbf{O}_2 e \mathbf{X} , onde a linha que liga os centros ópticos das câmeras é chamada de baseline . Sua interseção com os planos das imagens formam as linhas epipolares.	39
Figura 16 – Modelo completo de reconstrução 3D com <i>tracking</i> da câmera para extração de parâmetros da câmera. O escopo do trabalho está destacado pelo tracejado vermelho.	40
Figura 17 – Mapa de profundidade do <i>ground truth</i>	41
Figura 18 – Representação da cena por um conjunto de <i>voxels</i> para a etapa de reconstrução discreta do DTAM. São utilizadas uma imagem de referência I_r e um conjunto \mathcal{I} de M imagens I_m para tal.	42
Figura 19 – (a) <i>Ground truth</i> e mapas de profundidade da estimacão discreta (b) e contínua (c) do sistema DTAM. Foram utilizadas 30 imagens com $N = 32$ níveis de profundidade. O erro médio absoluto em relação ao <i>ground truth</i> de (b) foi de 24,27 <i>cm</i> e o de (c) foi de 23,09 <i>cm</i> . O percentual total de pixels reconstruídos em ambas é de 100 % pela reconstrução do DTAM ser de todos os pixels da imagem de referência.	44
Figura 20 – Nuvem de pontos do resultado apresentado na Figura 19b (a) e na Figura 19c (b).	44
Figura 21 – Esquemático resumido do algoritmo de reconstrução 3D esparsa.	45
Figura 22 – Imagem de referência I_r de um conjunto de imagens do banco de dados utilizado.	45
Figura 23 – Esquemático resumido dos algoritmos de tratamento de regiões homogêneas. (a) algoritmo que aproxima cada região segmentada por um plano. (b) algoritmo que aproxima uma superfície por meio da triangulação de Delaunay dentro de cada região.	46
Figura 24 – (a) processo de correspondência utilizando SAD para um pixel. (b) processo de correspondência utilizando SAD com janela de 3×3 pixels.	48
Figura 25 – (a) <i>Ground truth</i> e mapas de profundidade da estimacão utilizando SAD pixel a pixel (b) e janelas de 7×7 pixels (c). O erro médio absoluto em relação ao <i>ground truth</i> de (b) foi de 81,00 <i>cm</i> e o de (c) foi de 28,55 <i>cm</i> . A área preta que compreende o canto superior esquerdo representa os pixels que não foram reconstruídos por serem pontos da cena que não estavam na imagem 2.	49

Figura 26 – Exemplo de duas possíveis correspondências de W_1 em I_2 : W_{2a} e W_{2b} . Métrica NCC de W_{2a} e W_{2b} em relação a W_1 é de 0,92 e 0,37, respectivamente.	51
Figura 27 – (a) imagem I_1 . (b) imagem I_2 . (c) imagem retificada I_{1R} . (d) imagem retificada I_{2R} . As linhas epipolares estão representadas em vermelho.	52
Figura 28 – (a) plano da imagem com epipolo e e linhas epipolares em vermelho. (b) plano da imagem retificado com linhas epipolares horizontais e epipolo no infinito do eixo horizontal.	53
Figura 29 – Diferentes arranjos de sistemas compostos por duas câmeras. (a) Disposições onde os epipolos não se encontram dentro da imagem. (b) Disposições onde os epipolos se encontram dentro da imagem. Linhas epipolares estão em vermelho.	54
Figura 30 – Os novos eixos X , Y e Z para cada câmera estão apresentados como V_x , V_y e V_z , respectivamente.	54
Figura 31 – Sistema de imagens estéreo após a retificação. Planos da imagem paralelos a <i>baseline</i>	56
Figura 32 – Sistema de imagens estéreo após a retificação para $f_{ur} \neq f_{um}$. Ao se aplicar uma transformação de <i>zoom</i> (representada pelas linhas tracejadas), ambos planos da imagem se encontram no mesmo plano, com distância dos centros ópticos de f_{ur}	57
Figura 33 – Delimitação da linha epipolar por z_{min} e z_{max} para o sistema estéreo original e z_{minR} e z_{maxR} para o sistema estéreo retificado.	59
Figura 34 – (a) <i>Ground truth</i> e mapas de profundidade da estimação utilizando SAD pixel a pixel (b) e janelas de 7×7 pixels (c). O erro médio absoluto em relação ao <i>ground truth</i> de (b) foi de 81,00 <i>cm</i> e o de (c) foi de 28,55 <i>cm</i> . A área preta que compreende o canto superior esquerdo representa os pixels que não foram reconstruídos por serem pontos da cena que não estavam na imagem 2.	60
Figura 35 – (a) a janela vermelha apresenta uma superfície não plana. (b) a janela azul apresenta uma descontinuidade da profundidade da cena na transição entre o abajur e os livros do fundo. (c) e (d) são casos onde a superfície compreendida pela janela nas imagens é paralela e inclinada ao plano retificado, respectivamente. (e) representação da janela ideal de (d), onde os pixels comparados (pontos pretos) coincidem.	61
Figura 36 – Exemplo de oclusão no par de imagens estéreo Tsukuba. A área apontada pela seta amarela em W_1 sofre oclusão pelo abajur laranja em W_2	61

Figura 37 – (a) <i>Ground truth</i> e mapas de profundidade da estimação utilizando NCC (janelas de 7×7 pixels) sem (b) e com (c) verificação da consistência esquerda-direita. O erro médio absoluto em relação ao <i>ground truth</i> de (b) foi de 22,37 cm e o de (c) foi de 6,56 cm. As áreas pretas representam pixels que não foram reconstruídos ou por serem pontos da cena que não estavam na imagem 2 ou por serem dados como inválidos. O percentual total de pixels reconstruídos em (b) foi de 83 % e em (c) de 66 %	62
Figura 38 – (a) <i>Ground truth</i> e mapas de profundidade da estimação utilizando NCC com verificação esquerda-direita (janelas de 7×7 pixels), sem (b) e com (c) verificação de suspixels. O erro médio absoluto em relação ao <i>ground truth</i> de (b) foi de 6,56 cm e o de (c) foi de 5,66 cm. O percentual total de pixels reconstruídos em (b) foi de 66 % e em (c) de 60 %	64
Figura 39 – Nuvem de pontos referente a z_S para o caso sem (a) e com (a) verificação de suspixels.	64
Figura 40 – Nuvem de pontos do <i>ground truth</i> (a) e do resultado apresentado na Figura 38c (b).	65
Figura 41 – Aproximação local da função de métrica por uma parábola. Os três pontos que formam a parábola são o pixel retornado pela técnica <i>winner-takes-all</i> e seus dois vizinhos.	65
Figura 42 – Nuvem de pontos do resultado apresentado na Figura 38c antes (a) e depois (b) de se realizar a aproximação local da função de métrica por uma parábola. O erro médio absoluto em relação ao <i>ground truth</i> de (a) foi de 5,66 cm e o de (b) foi de 5,46 cm.	66
Figura 43 – Esquemático do processo descrito pelas Seções 3.3.1, 3.3.2 e 3.3.3. . . .	67
Figura 44 – Abordagem <i>coarse-to-fine</i> utilizando pirâmides gaussianas. O número P de níveis i nesta figura é igual a 3.	67
Figura 45 – Sobreamostragem de z_r para o processo de <i>matching</i> do próximo nível da pirâmide.	68
Figura 46 – Busca pela linha epipolar. No primeiro nível se busca por toda a linha epipolar, definida de z_{min} a z_{max} . Nos níveis posteriores, a busca é feita somente na vizinhança do correspondente indicado pelo nível anterior. As setas apontam o pixel no nível corrente relativo ao resultado do nível anterior (u_m^i). Os pixels em vermelho apresentam a linha epipolar de z_{min} a z_{max} e os de azul o resultado do <i>matching</i> no nível corrente. Os pontos pretos apresentam a linha epipolar definida pelos limites de cada nível.	69
Figura 47 – Esquemático do processo descrito pelas Seções 3.3.1, 3.3.2, 3.3.3 e 3.3.4 com a abordagem de pirâmide incorporada.	70

Figura 48 – funções de métrica SAD provenientes de 4 diferentes pares de imagem (I_r, I_m) sobre a linha epipolar (de u_1 a u_9) de um mesmo pixel \mathbf{u}_r . Cada par de imagens possui certo ruído na intensidade dos seus pixels, um dos motivos pelo qual as curvas não coincidem. Funções com ruído (a) fraco e (b) forte (gráficos superiores) e a média dessas funções em roxo (gráficos inferiores), resultando no correspondente u_{DTAM} , sendo u_5 a correspondência correta.	71
Figura 49 – Exemplo de resultado da consistência temporal u_t para os valores da função de métrica apresentados na Figura 48b. Nesse exemplo se desconsidera o fato de que são necessárias pelo menos 5 hipóteses semelhantes para se ter um conjunto válido. O quadrado azul apresenta o resultado da consistência temporal u_t , e o roxo apresenta o resultado u_{DTAM} do DTAM dado pelo exemplo da Figura 48b.	73
Figura 50 – (a) Mapa de profundidade pós etapa de consistência temporal para métrica NCC com verificação esquerda-direita (janelas de 7×7 pixels) com verificação de suspixels, e com aproximação local por parábola. (b) Nuvem de pontos do resultado de (a). O erro médio absoluto em relação ao <i>ground truth</i> foi de 2,35 cm contra 5,46 cm da Figura 42b, com percentual total de pixels reconstruídos de 60 %.	74
Figura 51 – Mapa de profundidade (a) e nuvem de pontos (b) do resultado z_S retornado pelo filtro de mediana de 5×5 pixels sobre o resultado apresentado na Figura 50. O erro médio absoluto em relação ao <i>ground truth</i> foi de 1,84 cm contra 2,35 cm da Figura 50.	75
Figura 52 – Esquemático do algoritmo de reconstrução 3D esparsa descrito pelas Seções 3.3.1 a 3.3.5.	75
Figura 53 – Visão de cima (a), de lado (b) e isométrica de uma calota azul.	76
Figura 54 – Exemplo do processo de limiarização do algoritmo MSER para a busca por regiões extremas ($\Delta = 85$). (a) Imagem em escala de cinza. Resultados da limiarização para (b) $\theta = 0$, (c) $\theta = 85$, (d) $\theta = 170$ e (e) $\theta = 255$	78
Figura 55 – A imagem (a) foi segmentada utilizando a técnica MSER no Matlab® (b), cujos parâmetros de configuração foram ' <i>ThresholdDelta</i> ' = 2 (Δ), ' <i>RegionAreaRange</i> ' = [30 307200], ' <i>MaxAreaVariation</i> ' = 0.25 e ' <i>ROI</i> ' = [1 1 480 640].	79
Figura 56 – Resultado da segmentação do método do MSER, onde a região A mostrada em (a) está contida na região B apresentada em (b).	80
Figura 57 – Resultado da segmentação da imagem mostrada na Figura 55b, onde cada cor representa uma região diferente.	80

Figura 58 – A imagem RGB I (a) foi segmentada utilizando a técnica SRM, resultando em I^* (b), cujo parâmetro de configuração $Q = 256$	81
Figura 59 – Exemplo do funcionamento do SRM para um vetor de 10 pixels, considerando que o critério de agrupamento é dado pela verificação de um limiar $\tau = 10$ em relação à média de cada região. Os vetores numéricos indicam a intensidade dos pixels do vetor de cores adjacente.	82
Figura 60 – A imagem RGB I_r (a) foi segmentada utilizando a técnica SRM, cujo parâmetro de configuração Q foi escolhido igual a 2 (b), 32 (c), 256 (d) e 1024 (e). O resultado da segmentação é melhor visualizado quando cada região corresponde a uma cor diferente, portanto, (f), (g), (h) e (i) apresentam esse resultado para as segmentações de (b), (c), (d) e (e), respectivamente.	83
Figura 61 – Resultado da segmentação da imagem mostrada na Figura 60a para $Q = 256$, onde cada cor representa uma região diferente.	83
Figura 62 – Grafo formado por vértices $v_i \in V$ e arestas $(v_i, v_j) \in E$. $w((v_i, v_j))$ indica o peso da aresta (v_i, v_j)	84
Figura 63 – Exemplo de <i>minimum spanning tree</i> para um componente C . As arestas que formam a <i>spanning tree</i> estão representadas de preto, e as demais do componente C de cinza. Tanto nesse exemplo quanto na técnica de Superpixels proposta por Felzenszwalb e Huttenlocher (2004), G é um grafo não direcional, ou seja, não existe um sentido único possível de se ir entre elementos.	85
Figura 64 – A imagem RGB I_r (a) foi segmentada utilizando a técnica de Superpixels, cujo parâmetro de configuração k foi escolhido igual a 300 (b), 500 (c), 700 (d) e 1000 (e). No resultados da segmentação, cada região corresponde a uma cor diferente.	86
Figura 65 – Resultado da segmentação da imagem mostrada na Figura 64b para $k = 300$, onde cada cor representa uma região diferente.	87
Figura 66 – Resultado da segmentação da imagem apresentada na Figura 55a pelos métodos MSER (a), SRM (b) e Superpixels (c).	88
Figura 67 – Matriz I_L contendo as regiões segmentadas, onde cada cor representa uma região diferente.	88
Figura 68 – Esquemático resumido dos algoritmos de tratamento de regiões homogêneas e densificação da nuvem de pontos. (a) algoritmo que aproxima cada região segmentada por um plano. (b) algoritmo que aproxima uma superfície por meio da triangulação de Delaunay dentro de cada região.	89
Figura 69 – Esquemático resumido do algoritmo de tratamento de regiões homogêneas aproximando cada região segmentada por um plano.	90

Figura 70 – Mapa de profundidade (a) e nuvem de pontos (b) resultante do processo de aproximação planar para o resultado apresentado na Figura 51. O erro médio absoluto em relação ao <i>ground truth</i> foi de 3,54 <i>cm</i> contra 1,84 <i>cm</i> , sendo o percentual total de pixels reconstruídos de 96 % contra 60 %.	92
Figura 71 – (a) Pontos a serem transformados em triângulos pela triangulação de Delaunay. (b) Conjunto de triângulos que não formam uma triangulação de Delaunay. (c) Conjunto de triângulos que formam uma triangulação de Delaunay. (d) Triângulos resultantes do processo de triangulação de Delaunay.	93
Figura 72 – (a) Triangulação de Delaunay realizada em um conjunto de pontos e (b) o resultado da superfície ao adicionar o conhecimento da profundidade de cada um dos vértices dos triângulos.	93
Figura 73 – Esquemático resumido do algoritmo de tratamento de regiões homogêneas aproximando cada região segmentada por uma superfície por meio da triangulação de Delaunay.	94
Figura 74 – Histograma dos valores de profundidade (b) da região destacada em branco (a). Os círculos vermelhos demarcam a região das ocorrências dadas como <i>inliers</i> , que determina os pontos utilizados para criar os triângulos (c), destacados em branco.	95
Figura 75 – Mapa de profundidade (a) e nuvem de pontos (b) resultante do processo de aproximação planar para o resultado apresentado na Figura 51. O erro médio absoluto em relação ao <i>ground truth</i> foi de 1,65 <i>cm</i> contra 1,84 <i>cm</i> , sendo o percentual total de pixels reconstruídos de 77 % contra 60 %.	96
Figura 76 – Imagem sintética foto-realista utilizada para os experimentos disponibilizada por Handa et al. (2012). Imagem de referência do conjunto de imagens.	97
Figura 77 – Mapa de profundidade do <i>ground truth</i> da imagem de referência.	98
Figura 78 – Erro absoluto do experimento do algoritmo para a métrica SAD sem a utilização de pirâmide sem (a) e com (b) verificação da consistência esquerda-direita (SAD3 e SAD4, respectivamente). A barra à direita da imagem indica a cor correspondente aos valores de erro absoluto em relação a <i>ground truth</i> em <i>cm</i> . Um pixel mal estimado em (b) que não foi descartado está indicado por uma seta vermelha.	102
Figura 79 – Sobreposição máxima entre as imagens e a imagem de referência (cor branca).	103

Figura 80 – Mapas de profundidade dos melhores resultados da métricas SAD2 (a), SSD2 (c) e FNCC2 (e) e suas respectivas nuvens de pontos (b,d,f) para a etapa de reconstrução 3D esparsa.	105
Figura 81 – Mapas de profundidade dos resultados das métricas SAD2 (a) e FNCC2 (c) para a etapa de reconstrução 3D densa por aproximação de planos e suas respectivas nuvens de pontos (b,d).	108
Figura 82 – Mapas de profundidade dos resultados das métricas SAD2 (a) e FNCC2 (c) para a etapa de reconstrução 3D densa por aproximação de superfície por Delaunay e suas respectivas nuvens de pontos (b,d).	109
Figura 83 – Mapas de profundidade da Rec. inicial (a) e Rec. final (c) do DTAM para 30 imagens e $N = 32$ níveis de profundidade, e suas respectivas nuvens de pontos (b,d).	111
Figura 84 – Imagem de referência do novo conjunto de 11 imagens (a) e sua segmentação por SRM (b). Conjunto esse que apresenta um outro ângulo da sala cena apresentada no conjunto de imagens utilizado para a obtenção dos resultados do Capítulo 4.	112
Figura 85 – Nuvens de pontos apresentadas em conjunto dos resultados de SAD com verificação de LRC e utilizando pirâmide gaussiana para as imagens de referência das Figuras 76 e 84. Em (a) foi utilizada a densificação da nuvem por aproximação por planos, em (b) por Delaunay.	113

Lista de tabelas

Tabela 1	– Resultados da reconstrução esparsa para as métricas SAD, SSD e FNCC.	100
Tabela 2	– Resultados da reconstrução 3D com aproximação por planos para as métricas SAD, SSD e FNCC.	106
Tabela 3	– Resultados da reconstrução 3D com aproximação por superfície utilizando triangulação de Delaunay para as métricas SAD, SSD e FNCC. .	107
Tabela 4	– Dois resultados de reconstrução 3D densa que se destacaram na Seção 4.2 e resultados da técnica do DTAM.	110

Lista de abreviaturas e siglas

CCD	Dispositivo de Carga Acoplada (<i>Charge-Coupled Device</i>)
DTAM	Rastreamento e Mapeamento Denso (<i>Dense Tracking And Mapping</i>)
FNCC	<i>Fast Normalized Cross Correlation</i>
MSER	Regiões Extremas Maximamente estáveis (<i>Maximally Stable Extremal Regions</i>)
NCC	<i>Normalized Cross Correlation</i>
PMVS	Reconstrução 3D com Múltiplas imagens baseada em Fragmentos da imagem (<i>Patch-based Multiview Stereo</i>)
RANSAC	Consenso de Amostras Aleatórias (<i>RANdom SAmple Consensus</i>)
RGB	Canais de cor Vermelho-Verde-Azul (<i>Red-Green-Blue</i>)
RGB-D	Canais de cor Vermelho-Verde-Azul e canal de profundidade da cena (<i>Red-Green-Blue-Depth</i>)
SAD	Soma das Diferenças Absolutas (<i>Sum of Absolute Differences</i>)
SIFT	Transformada de Características Invariantes a Rotação (<i>Scale-Invariant Feature Transform</i>)
SLIC	Clusterização Iterativa Linear Simples (<i>Simple Linear Iterative Clustering</i>)
SPRM	Agrupamento de regiões em Superpixels (<i>Superpixels Region Merging</i>)
SRM	Agrupamento de Regiões Estatísticas (<i>Statistical Region Merging</i>)
SSD	Soma das Diferenças Quadráticas (<i>Sum of Squared Differences</i>)
SURF	Extrator Robusto e Acelerado de Características (<i>Speeded Up Robust Features</i>)
TMVS	Reconstrução 3D com Múltiplas imagens baseada em Tensores (<i>Tensor-based Multiview Stereo</i>)
VANTs	Veículos Aéreos Não Tripulados

Lista de símbolos

S_1	Sistema de coordenadas da câmera 1
X_1, Y_1, Z_1	Eixos de coordenadas métricos da câmera 1
x_1, y_1, z_1	Coordenadas métricas da câmera 1
\mathbf{X}_1	Ponto tridimensional da câmera 1 em metros, $\mathbf{X}_1 = (x_1, y_1, z_1)^T$
$\dot{\mathbf{X}}_1$	Ponto \mathbf{X}_1 em coordenadas homogêneas, $\dot{\mathbf{X}}_1 = (x_1, y_1, z_1, 1)^T$
u, v	Eixos e coordenadas da imagem em pixels (horizontal e vertical, respectivamente)
\mathbf{u}_1	Ponto bidimensional da imagem da câmera 1 em pixels, $\mathbf{u}_1 = (u_1, v_1)^T$
$\dot{\mathbf{u}}_1$	Ponto \mathbf{u}_1 em coordenadas homogêneas, $\dot{\mathbf{u}}_1 = (u_1, v_1, 1)^T$
f	Distância focal em unidades métricas
s_u, s_v	número de pixels por unidade de comprimento do sensor (horizontal e vertical, respectivamente), $f_u = f s_u$ e $f_v = f s_v$
\mathbf{c}_o	Ponto principal da câmera, $\mathbf{c}_o = (u_o, v_o)$
K_1	Matriz de parâmetros intrínsecos da câmera 1
R_{21}	Matriz de rotação do referencial 1 para 2
t_{21}	Matriz de translação do referencial 1 para 2
T_{21}	Matriz de transformação (R_{21}, t_{21})
\mathbf{O}_1	Centro óptico da câmera 1
\mathbf{e}_1	Epipolo 1, que é a projeção do centro óptico da câmera 2 (\mathbf{O}_2) no plano da imagem da câmera 1
S_w	Sistema de coordenadas global
\mathbf{O}_{1w}	Centro óptico da câmera 1 no referencial global
I_r	Imagem de referência
I_m	Imagem m
M	Quantidade de pares (I_r, I_m) de imagens RGB utilizados

\mathcal{I}	Conjunto de M imagens RGB
$I_r(v, u)$	Intensidade RGB do pixel (u, v) na imagem I_r
$I_r(v, u, l)$	Intensidade do canal l do pixel (u, v) na imagem I_r
W_r, W_m	Janela RGB no entorno dos pixels \mathbf{u}_r e \mathbf{u}_m , respectivamente
\overline{W}	Vetor com as médias das intensidades dos três canais de cor da janela W
L	Quantidade de linhas nas imagens
C	Quantidade de colunas nas imagens
N	Quantidade de níveis discretos de profundidade
z_r	Profundidade estimada dos pixels da imagem I_r
P	Quantidade de níveis da pirâmide gaussiana
I_r^i, I_m^i, z_r^i	Variáveis referentes ao nível i da pirâmide gaussiana
I_{rR}^i, I_{mR}^i	Variáveis referentes ao sistema retificado
$\mathbf{u}_{rR}, \mathbf{u}_{mR}$	Variáveis referentes ao sistema retificado
z_R	Profundidade estimada dos pixels de I_r no sistema retificado, onde $z_R = z_r R = z_m R$
V_x, V_y, V_z	Eixos de coordenadas métricos da câmera 1
\vec{b}	Baseline, $\vec{b} = \mathbf{O}_{2w} - \mathbf{O}_{1w}$
$z_{r[k]}$	Profundidade estimada do par (I_r, I_k)
$z_{r[1-10]}$	Conjunto das hipóteses de profundidade dos 10 pares (I_r, I_m)
z_t	Resultado da fusão das hipóteses $z_{r[1-10]}$
z_S	Resultado de estimação de profundidade esparsa
I_L	Matriz contendo os rótulos de cada região segmentada pelo SRM na imagem I_r
z_D	Resultado de estimação de profundidade densa

Sumário

1	INTRODUÇÃO	22
1.1	Estado da arte	23
1.2	Problema abordado nesta dissertação	26
1.3	Objetivo	27
1.4	Resumo da abordagem adotada para a solução do problema	27
1.5	Estrutura do texto	28
2	MODELO DE CÂMERA E SISTEMA MULTICÂMERAS	30
2.1	Modelo de câmera	30
2.2	Caso de múltiplas câmeras	35
2.3	Geometria epipolar	38
3	MÉTODO PROPOSTO	40
3.1	Dense Tracking And Mapping (DTAM)	41
3.2	Visão geral da abordagem proposta	43
3.3	Reconstrução 3D esparsa	47
3.3.1	<i>Matching</i>	47
3.3.2	Retificação epipolar	50
3.3.3	Oclusões, suspixels e precisão sub-pixel	59
3.3.4	Pirâmide gaussiana	66
3.3.5	Filtragem dos resultados	70
3.4	Segmentação de regiões homogêneas para densificar a nuvem de pontos	75
3.4.1	<i>Maximally Stable Extremal Regions</i> (MSER)	77
3.4.2	<i>Statistical Region Merging</i> (SRM)	80
3.4.3	Supapixel	84
3.4.4	Comparação entre as técnicas de segmentação de imagens	87
3.5	Reconstrução 3D densa	89
3.5.1	Primeira abordagem: cada região é um plano	89
3.5.2	Segunda abordagem: cada região é uma superfície	92
4	EXPERIMENTOS E RESULTADOS	97
4.1	Reconstrução 3D esparsa	98
4.2	Reconstrução 3D densa	104
4.3	Comparação dos resultados com o DTAM	110
4.4	Ampliando reconstruções prévias	112

5	CONCLUSÕES E TRABALHOS FUTUROS	115
	REFERÊNCIAS	117

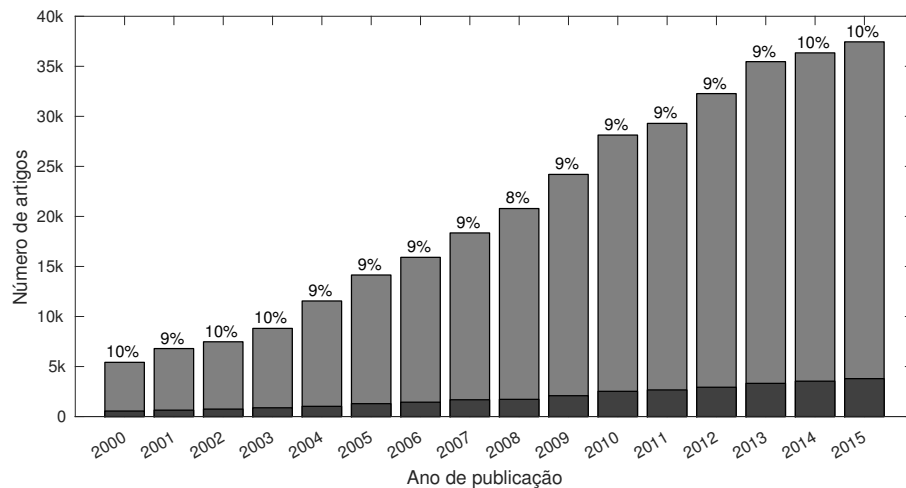
1 Introdução

Câmeras digitais estão entre os sensores mais utilizados, principalmente devido à riqueza de informações contidas em uma imagem e ao avanço tecnológico, que permitiu o seu barateamento. Assim, câmeras tornaram-se mais comuns em sistemas embarcados, celulares, computadores, etc, podendo ser usadas, em muitos casos, para substituir sensores laser, que normalmente são mais caros que câmeras.

Além disso, com o aumento da capacidade de processamento dos dispositivos, imagens podem ser usadas para analisar cenas e extrair informações tridimensionais dos pixels em um espaço de tempo bem menor do que se conseguia fazer há alguns anos atrás. Esse processo de recuperação da informação tridimensional do meio recebe o nome de reconstrução 3D.

A quantidade de trabalhos sobre o assunto cresceu consideravelmente nas últimas décadas. A Figura 1 apresenta um gráfico comparativo obtido utilizando a plataforma Scopus (2016), que possui uma vasta base de periódicos e livros para análises como essa.

Figura 1 – Quantidade de trabalhos que abordam o tema de reconstrução 3D (cinza escuro) dentro da área de visão computacional (cinza claro). Esse quantia também está apresentada em percentual da área de visão computacional.



Fonte: Produção do próprio autor.

Como pode-se notar na Figura 1, nos últimos anos houve um grande crescimento na área de visão computacional. O número de artigos sobre reconstrução 3D acompanha esse crescimento de forma percentualmente constante. Ou seja, reconstrução 3D é um assunto que cresce dentro de uma área em constante processo de inovação. Dessa forma, o número de trabalhos e algoritmos que abordam o tema de reconstrução hoje é muito

grande, com diversas propostas de solução. Alguns desses trabalhos serão comentados e discutidos a seguir.

1.1 Estado da arte

Como comentado anteriormente, com o aumento da capacidade de processamento dos dispositivos, imagens podem ser usadas para analisar cenas e extrair informações tridimensionais dos pixels em um espaço de tempo bem menor do que se conseguia fazer há alguns anos atrás.

Mesmo assim, obter uma boa reconstrução 3D do ambiente ainda é um processo computacionalmente custoso. Isso é um problema que a área de sistemas embarcados enfrenta, já que, geralmente, o tempo de processamento das informações deve ser baixo para que os sistemas consigam responder rapidamente (e.g. aplicações de tempo real). Por isso, muitos trabalhos tentam equilibrar os ganhos em velocidade de processamento e qualidade da reconstrução 3D.

Newcombe, Lovegrove e Davison (2011) propõem um sistema chamado *Dense Tracking And Mapping* (DTAM) que, em tempo real, faz a aquisição de imagens com uma câmera em movimento, estima a sua posição corrente e realiza a reconstrução 3D da cena filmada (Figura 2a). Por outro lado, Wu et al. (2010) realiza a reconstrução de imagens salvas em seu repositório, com as posições já estimadas (câmera calibrada), por meio de uma técnica com alto custo computacional chamada *Tensor-based Multiview Stereo* (TMVS) (Figura 2a).

Figura 2 – (a) Reconstrução 3D do método DTAM. (b) Reconstrução 3D da técnica TMVS.



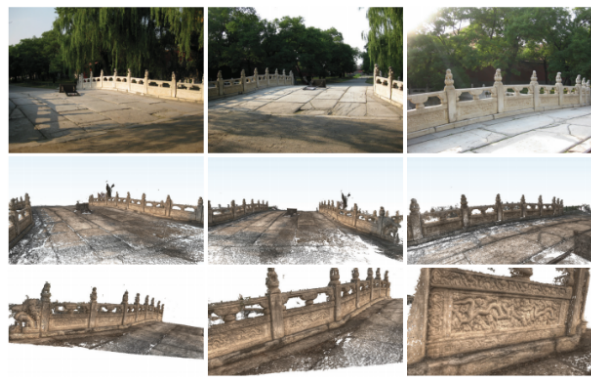
Fonte: (a) Newcombe, Lovegrove e Davison (2011). (b) Wu et al. (2010).

A diferença entre essas duas técnicas está no equilíbrio entre velocidade e qualidade.

Enquanto Newcombe, Lovegrove e Davison (2011) deram mais importância ao tempo de processamento de seu sistema, Wu et al. (2010) escolheu ter uma maior precisão na reconstrução 3D em seus testes. Essa decisão depende das características do problema abordado. Uma técnica que necessita responder a uma situação rapidamente, realizando a reconstrução em tempo de execução, é classificada como método *online*, já as técnicas que não processam as imagens no momento da aquisição, não sendo obrigadas a responder a especificações de tempo baixas, são classificadas como métodos *offline*.

A mesma escolha por qualidade pode ser vista em (PAGANI et al., 2011). Esse trabalho criou uma variante da técnica *offline* chamada *Patch-based Multiview Stereo* (PMVS), proposta por Furukawa e Ponce (2010) para imagens em perspectiva. Essa variante foi desenvolvida para ser utilizada em imagens esféricas de altíssima resolução para a estimativa da profundidade de ambientes (Figura 3).

Figura 3 – Reconstrução 3D da técnica PMVS.



Fonte: Pagani et al. (2011).

Um método de reconstrução também pode ser classificado quanto à distância entre as câmeras. Uma técnica de reconstrução 3D para sistemas cuja a distância entre câmeras é alta caracteriza um método de *wide baseline* (STRECHA; TUYTELAARS; GOOL, 2003), do contrário, são classificadas como técnicas de *small baseline* (DIAS; ARAUJO; MIRALDO, 2016). Geralmente, métodos de *small baseline* são aplicados a câmeras em movimento, ou seja, na sequência de imagens de um vídeo (PRADEEP et al., 2013). O trabalho aqui proposto pode trabalhar com ambas situações, sendo baixa a precisão para reconstrução de pontos muito distantes por um sistema de câmeras de *small baseline* e maior por um sistema de *wide baseline*. Entretanto, é mais fácil embarcar em um robô, por exemplo, um sistema de *small baseline*, ou seja, a decisão de qual sistema usar também depende da aplicação.

Pesquisas, como (IZADI et al., 2011; BYLOW et al., 2013; NEWCOMBE; LOVEGROVE; DAVISON, 2011), chegam a atingir resultados de reconstrução 3D em tempo real. Izadi et al. (2011), Bylow et al. (2013) utilizam imagens RGB-D em seus algoritmos

para realizar a reconstrução, ou seja, imagens RGB e informação vinda de um sensor de profundidade (e.g. Kinect da Microsoft). Nesses sistemas, há uma fusão dos dados para se obter uma estimativa mais refinada de profundidade. No caso deste trabalho, somente imagens serão usadas como informação visual a priori, onde cada imagem será considerada como uma câmera posicionada em um local diferente.

Uma alternativa ao sistema de uma câmera em movimento é a utilização de um sistema estéreo, assim como o sistema ocular humano. Dessa forma, é possível estimar a profundidade da cena com apenas uma aquisição do par de imagens (na verdade, geralmente, se utiliza um conjunto de resultados de reconstrução para filtrar ruídos de estimação, como será mostrado no Capítulo 3). Alcantarilla, Beall e Dellaert (2013) apresentam um sistema estéreo que realiza a reconstrução 3D densa enquanto se movimenta por um bairro (Figura 4). Seu algoritmo consegue estimar a profundidade de uma cena e lidar com problemas relacionados a objetos em movimento na cena e memória limitada para armazenar a quantidade de pontos reconstruídos (nuvem de pontos).

Figura 4 – Reconstrução 3D do sistema proposto por Alcantarilla, Beall e Dellaert (2013).



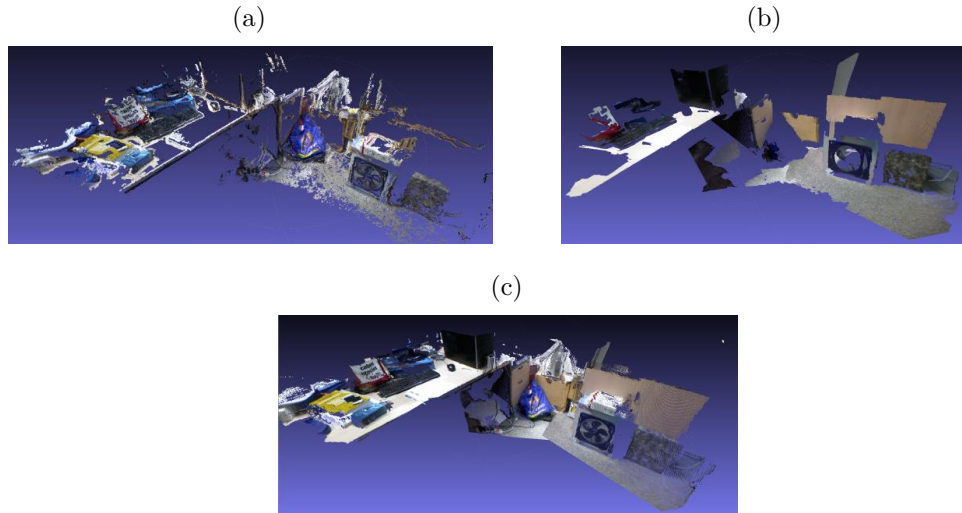
Fonte: Alcantarilla, Beall e Dellaert (2013).

Uma desvantagem dessa abordagem está na necessidade de se ter um sistema estéreo, algo que não é tão comum de se encontrar quanto um sistema monocular. Apesar de utilizar um par de imagens, o método aqui proposto pode realizar a reconstrução 3D a partir de imagens vindas de qualquer dispositivo: um celular ou um computador com câmera, uma *webcam*, etc. Ou seja, estima a profundidade da cena a partir de qualquer par de imagens.

Mesmo com muitos trabalhos abordando o tema de reconstrução, a estimação da profundidade em áreas com pouca textura é um problema não resolvido. Para regiões homogêneas, a ambiguidade é elevada, dificultando a correspondência entre pixels. Consequentemente, mesmo após o processo completo de reconstrução, a precisão nessas áreas acaba sendo muito prejudicada.

Esse problema é discutido em (CONCHA; CIVERA, 2015) e (CONCHA et al., 2015), que apresentam formas de se aumentar a precisão do DTAM em regiões homogêneas, utilizando técnicas que não prejudiquem muito a velocidade do processo de reconstrução 3D. Essas técnicas adicionam informações de entendimento da cena à etapa de estimativa densa

Figura 5 – Reconstrução 3D do sistema proposto por Concha e Civera (2015). (a) Reconstrução esparsa. (b) Aproximação planar de regiões homogêneas. (c) Resultado da junção de (a) e (b).



Fonte: Concha e Civera (2015).

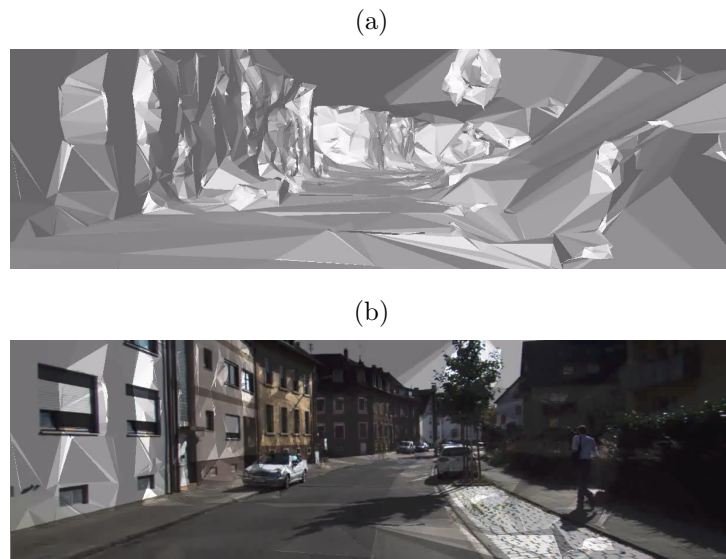
do DTAM como, por exemplo, aproximação planar de áreas homogêneas, e classificação de regiões das imagens como sendo parte do teto, das paredes, do chão dentre outras (Figura 5).

A consideração de planaridade em regiões homogêneas nem sempre pode ser feita, visto que algoritmos de segmentação podem encontrar resultados apresentando problemas de subsegmentação, o que poderia levar uma região a conter superfícies não planas ou até descontínuas. Para atender a esses casos, Romanoni e Matteucci (2015) constroem uma triangulação de Delaunay a partir de uma nuvem esparsa de pontos, possibilitando a reconstrução de superfícies não planas e quinas, ou seja, saltos de profundidade (Figura 6). Assim, é possível de forma *online* reconstruir cenas com o objetivo de auxiliar tarefas, por exemplo, de seguimento de trajetória, desvio de obstáculos e interação com o ambiente.

1.2 Problema abordado nesta dissertação

Hoje, o laboratório onde esta dissertação está sendo desenvolvida tem uma série de trabalhos que podem se beneficiar de um processo de reconstrução como o Espaço Inteligente (RAMPINELLI et al., 2014; QUEIROZ et al., 2015; RAMPINELLI et al., 2015), projetos com veículos aéreos não tripulados (VANTs) (AMORIM et al., 2015a; AMORIM et al., 2015b), navegação de robôs (SA et al., 2014) e interação homem-robô (PEREIRA; VASSALLO; SALLES, 2013), ou seja, estudar e fazer reconstrução 3D representa um tópico de forte interesse e aplicação para o atual grupo de pesquisa.

Figura 6 – Reconstrução 3D do sistema proposto por Romanoni e Matteucci (2015). Reconstrução utilizando triangulação de Delaunay com (b) e sem (a) textura.



Fonte: Romanoni e Matteucci (2015).

Então, o problema abordado nessa dissertação pode ser descrito como obter a reconstrução 3D de uma cena a partir de imagens capturadas durante um movimento genérico (que inclui translação e rotação) a fim de obter uma nuvem de pontos ou aproximação de superfícies para que isso seja usado em outros trabalhos do laboratório.

1.3 Objetivo

O objetivo, de forma sucinta, é obter um método de reconstrução que represente um bom compromisso entre precisão e velocidade em aplicações dentro do laboratório.

Vale ressaltar que o foco do trabalho está apenas na etapa de reconstrução, pois o processo de estimativa dos parâmetros intrínsecos e extrínsecos não faz parte do escopo e serão considerados já obtidos através de métodos de calibração ou estimativas de “*egomotion*” (recuperação do movimento da câmera).

Como já mencionado, será considerado um movimento genérico, ou seja, que não seja preferencialmente em linha reta, devida a limitações que isto traz para o processo de reconstrução. Limitações que serão discutidas no decorrer desta dissertação.

1.4 Resumo da abordagem adotada para a solução do problema

O método proposto nessa dissertação é composto por uma etapa de reconstrução 3D esparsa e um pós-processamento para tratar regiões com pouca textura com o objetivo

de se obter uma nuvem de pontos mais densa.

Na etapa de reconstrução esparsa, é utilizado um conjunto de 10 imagens que possuem certa sobreposição com uma imagem de referência. Cada par (imagem, imagem de referência) é retificado. Em seguida, os pixels da imagem de referência são buscados nas 10 imagens comparando-se possíveis correspondentes por meio de uma métrica de similaridade. Para diminuir o tempo de processamento do algoritmo, utilizou-se uma pirâmide gaussiana de multiresolução.

Uma vez encontrados pares de pontos correspondentes em diferentes imagens, é possível calcular a profundidade dos pixels da imagem de referência. Os 10 pares resultam em 10 hipóteses de profundidade por pixel. Esses resultados são fundidos e filtrados por duas etapas, filtro temporal e filtro espacial, onde são eliminados pixels com possíveis erros de correspondência (considerados como *outliers*).

Os pixels reconstruídos, após as etapas de filtragem, representam uma nuvem de pontos esparsa. Para que a reconstrução 3D torne-se densa, duas abordagens foram analisadas e propostas, aproximando-se regiões homogêneas por planos e por uma superfície a partir da triangulação de Delaunay. Nos dois casos, a imagem de referência é segmentada em regiões homogêneas de cor similar utilizando a técnica *Statistical Region Merging* (SRM).

Como se deseja obter um método de reconstrução que represente um bom compromisso entre precisão e velocidade, a metodologia proposta foi comparada à técnica DTAM que atingiu resultados em tempo real. Finalmente, os resultados mostram que a metodologia proposta conseguiu gerar uma reconstrução atendendo aos requisitos inicialmente traçados.

Durante as pesquisas realizadas nesta dissertação de Mestrado, o artigo “Aproximação planar por partes para reconstrução 3D densa” foi publicado no XXI Congresso Brasileiro de Automática em outubro de 2016 (SILVA; LUBE; VASSALLO, 2016). Mesmo assim, ainda será escrito um artigo de revista que reúna os novos resultados e análises realizados após a publicação do artigo citado e que foram obtidos até a conclusão deste trabalho.

1.5 Estrutura do texto

Esta Dissertação de Mestrado será apresentada em 5 capítulos. O Capítulo 1 apresenta a motivação para se trabalhar com o tema de reconstrução, contextualiza o problema abordado e descreve sucintamente algumas das técnicas atuais que lidam com reconstrução 3D. Além disso, nesse mesmo capítulo as contribuições e o objetivo deste trabalho são apresentados.

O Capítulo 2 explicita o modelo matemático utilizado para representar uma câmera e os parâmetros que relacionam os pontos tridimensionais da cena com os pixels de uma foto. Também se apresenta, nesse capítulo, a notação utilizada.

A abordagem adotada para a solução do problema é apresentada no Capítulo 3, sendo explicada cada etapa detalhadamente.

Os experimentos realizados para se avaliar a metodologia proposta e seus resultados são apresentados e discutidos no Capítulo 4. A conclusão, propondo alguns trabalhos futuros, é abordada no Capítulo 5.

2 Modelo de câmera e sistema multicâmeras

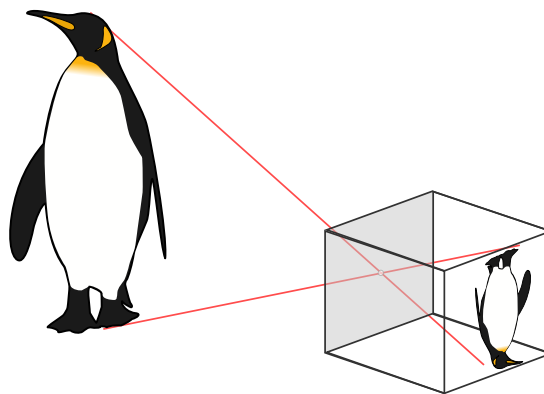
Sabe-se que as câmeras são muito utilizadas para sensoriamento em robótica devido à quantidade de informação possível de se extrair de imagens. Entretanto, quando se captura a foto de uma cena, perde-se a informação de profundidade do meio já que todo o ambiente fotografado fica registrado no CCD da câmera como uma projeção 2D da cena.

Neste capítulo, será explicitado o modelo matemático utilizado para representar uma câmera e os parâmetros que relacionam os pontos tridimensionais da cena com os pixels de uma foto.

2.1 Modelo de câmera

As câmeras são ferramentas que armazenam a informação visual de uma cena em uma matriz de pixels. Para que isso seja possível, a luz refletida pelo mundo deve alcançar uma matriz de sensores sensíveis à luz (CCD) no interior da câmera. O modelo de câmera mais utilizado para se entender o processo é o **modelo *pinhole***. Nesse modelo a luz deve passar através de um furo bem pequeno, formando uma imagem focada no fundo de uma câmara escura, conforme Figura 7.

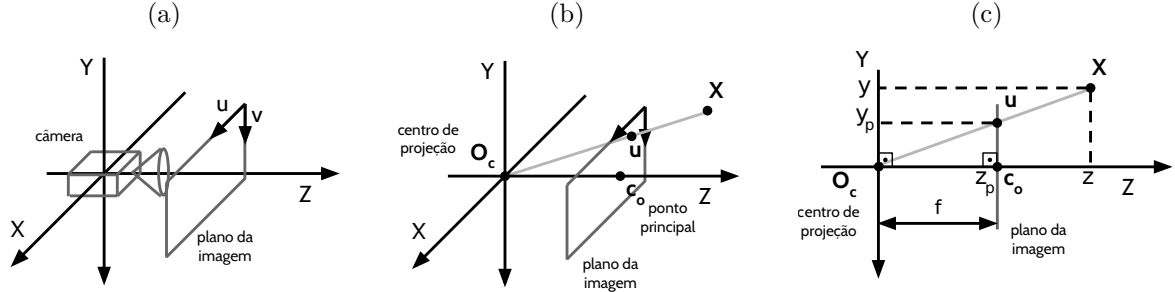
Figura 7 – Representação do registro de uma cena por uma câmera no modelo de câmera *pinhole*.



Fonte: Produção do próprio autor.

O modelo de câmera *pinhole* (HARTLEY; ZISSERMAN, 2003) é utilizado para representar esse processo matematicamente. Nessa representação, a cena é armazenada na forma de uma projeção do mundo em um plano chamado plano da imagem. A Figura 8 mostra o processo de tal representação, onde o centro óptico de projeção da câmera O_c é a origem do eixo de coordenadas da câmera. O plano da imagem é paralelo ao plano formado pelos eixos X e Y e perpendicular ao eixo Z . Além disso, ele está a uma distância

Figura 8 – O sistema de coordenadas da câmera (a) pode ser representado também como (b) onde o eixo Z intercepta o plano da imagem no ponto principal \mathbf{c}_o . Um ponto tridimensional \mathbf{X} é projetado no plano da imagem no pixel \mathbf{u} . (c) apresenta essa projeção pela vista do plano YZ .



Fonte: Produção do próprio autor.

focal f do centro óptico. O ponto onde o eixo Z intercepta o plano da imagem é chamado de ponto principal da câmera $\mathbf{c}_o = (u_o, v_o)$, sendo esse, comumente, o ponto central da imagem.

Dessa forma, é possível calcular o pixel $\mathbf{u} = (u, v)^T$ no qual o ponto $\mathbf{X} = (x, y, z)^T$ é projetado. Primeiramente, encontra-se seu ponto em coordenadas métricas $\mathbf{X}_p = (x_p, y_p, z_p)^T$ obtido na projeção de \mathbf{X} no plano da imagem, para depois relacionar o resultado para valores em pixels \mathbf{u} . Considerando que a distância focal é dada na unidade métrica, tem-se que

$$x_p = f \frac{x}{z}, \quad (2.1)$$

$$y_p = f \frac{y}{z} \quad (2.2)$$

e z_p assumindo valor igual à distância focal f para qualquer ponto projetado no plano da imagem.

Mesmo que tenha sido calculado o valor de z_p , representa-se um pixel como um ponto pertencente a uma matriz bidimensional. Desta forma, desconsidera-se o valor de z_p e encontra-se a quantia de x_p e y_p equivalente em pixels. Assim, a transformação que relaciona (x_p, y_p) na unidade métrica a seu valor (u_p, v_p) em pixels é dada por

$$u_p = s_u x_p = s_u f \frac{x}{z}, \quad (2.3)$$

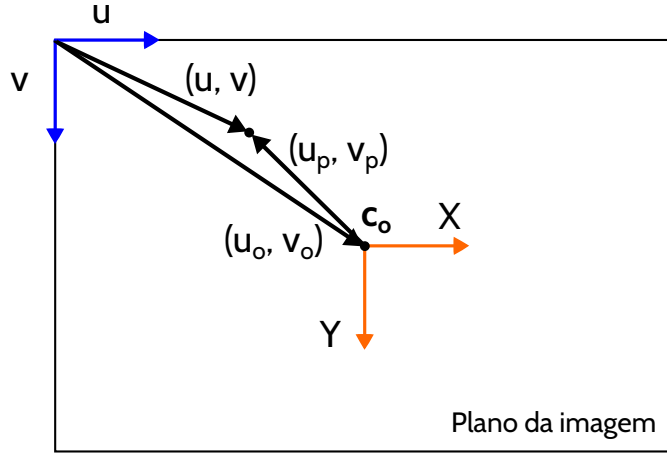
$$v_p = s_v y_p = s_v f \frac{y}{z}, \quad (2.4)$$

onde os parâmetros s_u e s_v dependem das dimensões dos sensores do CCD (em unidade métrica) ao longo dos eixos X e Y , respectivamente, e representam o número de pixels

por unidade de comprimento do sensor. Quando $s_u = s_v$ os pixels são quadrados. Caso contrário, os pixels são retangulares.

Ainda assim, $\mathbf{u}_p = (u_p, v_p)^T$ está dado num sistema centrado no ponto principal da câmera. Para que a referência seja tomada como o canto superior esquerdo da imagem, como normalmente se considera em uma matriz numérica, as coordenadas de \mathbf{u}_p devem ser transladadas, como apresentado na Figura 9.

Figura 9 – Translação da referência no plano da imagem do ponto principal \mathbf{c}_o para o canto superior esquerdo.



Fonte: Produção do próprio autor.

Matematicamente, tem-se que

$$u = u_p + u_o, \quad (2.5)$$

$$v = v_p + v_o, \quad (2.6)$$

onde $u_p = s_u f \frac{x}{z}$ e $v_p = s_v f \frac{y}{z}$.

Para simplificar a escrita, $s_u f$ e $s_v f$ serão representados, respectivamente, por f_u e f_v . Dessa maneira, as Equações 2.5 e 2.6 podem ser reescritas como

$$u = f_u \frac{x}{z} + u_o, \quad (2.7)$$

$$v = f_v \frac{y}{z} + v_o. \quad (2.8)$$

Se as Equações 2.7 e 2.8 forem organizadas matricialmente de forma a evidenciar os pontos \mathbf{u} e \mathbf{X} chega-se a

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_u & 0 & u_o \\ 0 & f_v & v_o \end{pmatrix} \begin{pmatrix} x/z \\ y/z \\ z/z \end{pmatrix}. \quad (2.9)$$

Percebe-se que todos os elementos do ponto \mathbf{X} estão divididos por z , de forma que para explicitar de fato o ponto \mathbf{X} , evidencia-se z como um fator de escala. Usualmente, se representa o pixel \mathbf{u} em sua forma de coordenadas homogêneas $\dot{\mathbf{u}} = (u, v, 1)^T$, assim, a Equação 2.9 se torna

$$z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} zu \\ zv \\ z \end{pmatrix} = \begin{pmatrix} f_u & 0 & u_o \\ 0 & f_v & v_o \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \quad (2.10)$$

Na Equação 2.10, a profundidade z já foi expressada como um fator de escala. Esse pode ser retirado ao se dividir o vetor resultante da projeção pelo seu terceiro elemento, visto que este é o próprio z . De forma compacta, a expressão pode ser representada por

$$z\dot{\mathbf{u}} = K\mathbf{X}, \quad (2.11)$$

onde

$$K = \begin{pmatrix} f_u & 0 & u_o \\ 0 & f_v & v_o \\ 0 & 0 & 1 \end{pmatrix}. \quad (2.12)$$

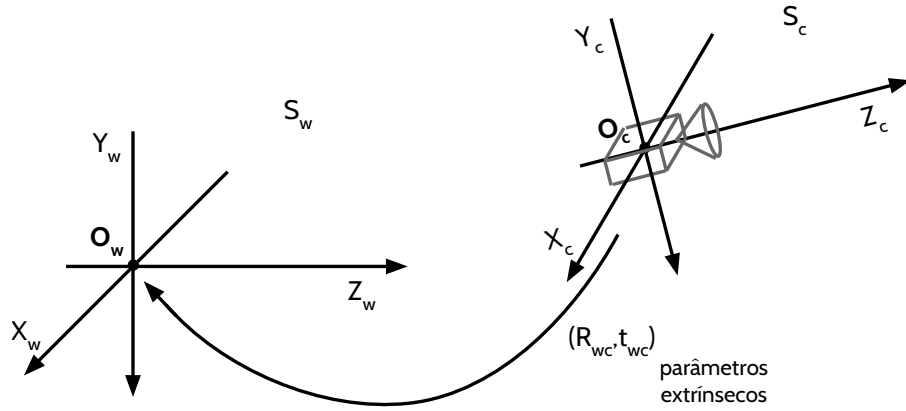
Essa matriz é conhecida como matriz de **parâmetros intrínsecos** da câmera, visto que esses parâmetros dependem de fatores construtivos de cada dispositivo.

Nessa representação, os pontos da cena já estão no sistema de coordenadas da câmera, não são considerados a distorção radial e o fator de cisalhamento entre os dois eixos do plano da imagem. O cisalhamento é normalmente considerado zero e a distorção radial pode ser tratada separadamente do modelo de projeção.

Além de se considerar as características intrínsecas da câmera, é comum em aplicações de reconstrução 3D, de localização de objetos e de localização e mapeamento simultâneo de robôs (SLAM), se considerar um sistema de coordenadas global onde o robô, o objeto ou a câmera estão inseridos. Dessa forma, precisa-se encontrar uma transformação de corpo rígido que associe esse referencial global (e.g. referencial do mundo) com os demais referenciais locais (e.g. referencial da câmera, do robô e do objeto).

As transformações de rotação $R_{wc} \in \mathbb{SO}(3)$ e translação t_{wc} que levam um ponto no sistema de coordenadas da câmera S_c para o sistema de coordenadas global S_w (Figura 10)

Figura 10 – Transformação (R_{wc}, t_{wc}) do sistema de coordenadas da câmera S_c para o global S_w .



Fonte: Produção do próprio autor.

recebem o nome de **parâmetros extrínsecos**, por dependerem de dados externos, e não internos à câmera.

Dessa forma, a transformação de um ponto no referencial 1 para o referencial 2 pode ser escrita como

$$\mathbf{X}_2 = R_{21}\mathbf{X}_1 + t_{21}. \quad (2.13)$$

A transformação (R_{21}, t_{21}) muitas vezes é representada pela matriz

$$T_{21} = \begin{pmatrix} R_{21} & t_{21} \\ 0^T & 1 \end{pmatrix}, \quad (2.14)$$

que considera \mathbf{X}_1 e \mathbf{X}_2 em coordenadas homogêneas $\dot{\mathbf{X}}_1 = (x_1, y_1, z_1, 1)^T$ e $\dot{\mathbf{X}}_2 = (x_2, y_2, z_2, 1)^T$, respectivamente. Assim, a Equação 2.13 passa a ser representada como

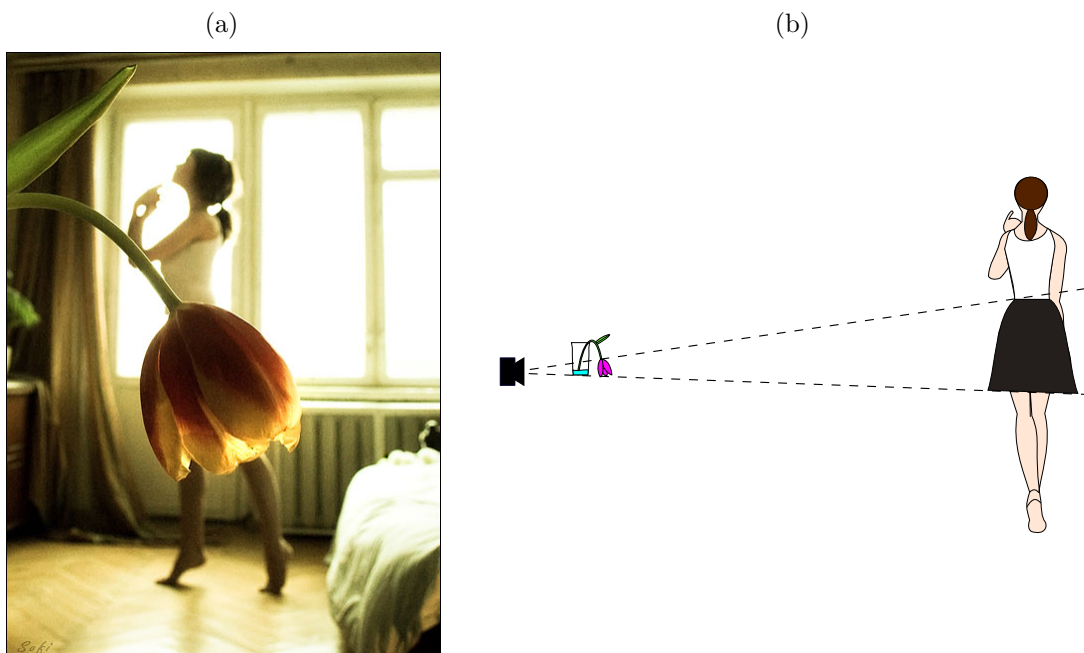
$$\dot{\mathbf{X}}_2 = T_{21}\dot{\mathbf{X}}_1 = T_{21} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}. \quad (2.15)$$

Tais parâmetros, intrínsecos e extrínsecos, podem ser obtidos por métodos de calibração de câmeras (HARTLEY; ZISSERMAN, 2003).

2.2 Caso de múltiplas câmeras

A maioria das aplicações de visão computacional, como é o caso abordado por este trabalho, necessita de mais do que somente uma imagem para obter a informação desejada. Fotos que brincam com o conceito de perspectiva são uma boa forma de se exemplificar isso. A Figura 11 apresenta essa ideia.

Figura 11 – A foto que brinca com o conceito de perspectiva (a) pode ser também conferida de outro ângulo (b) que mostra a localização da flor e da menina na sala onde a foto foi tirada. Em (b) pode-se notar por meio de linhas que a projeção da cena na matriz de sensores da câmera terá a flor com mesmo tamanho que a saia da menina.



Fonte: (a) Mikhina (2017). (b) Produção do próprio autor.

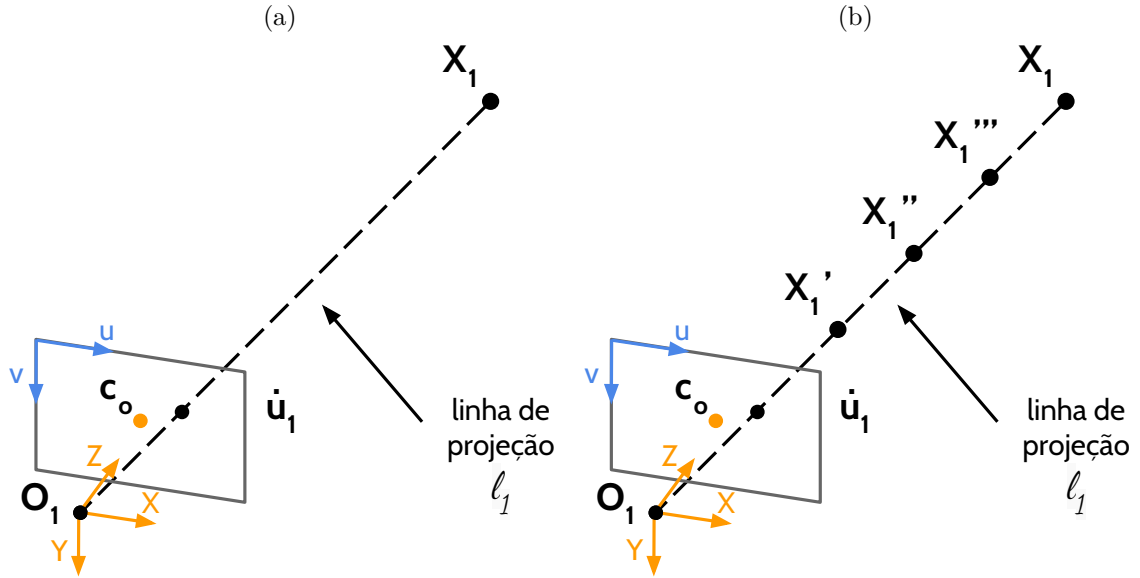
A Figura 11a apresenta uma flor que, na foto, parece a saia de uma moça. Como mostrado na Figura 11b, isso acontece pois a flor se encontra mais próxima da câmera que a menina, dando a impressão de que a flor tem proporções parecidas com o tamanho de saia que ela usaria. Assim, apesar da foto registrar somente a cor e o formato dos objetos, a profundidade deles influencia no tamanho que eles assumem na imagem.

No entanto, esse tipo de informação só é óbvia para um ser humano por ele conhecer as proporções de uma flor e de uma pessoa. Por outro lado, um robô não tem, a priori, tal informação. Para ele, a menina poderia ter $1,76\text{ m}$ de altura e estar longe, ou ter 30 cm e estar utilizando uma saia feita de flor.

É possível representar esse exemplo geometricamente. A Figura 12 apresenta novamente a projeção de um ponto tridimensional no plano da imagem no sistema de

coordenadas da câmera.

Figura 12 – (a) Projeção do ponto tridimensional \mathbf{X}_1 no plano da imagem da câmera \mathbf{O}_1 .
 (b) Os pontos $\mathbf{X}'_1, \mathbf{X}''_1$ e \mathbf{X}'''_1 sobre a linha ℓ_1 também possuem mesma projeção que \mathbf{X}_1 em \mathbf{u}_1 .



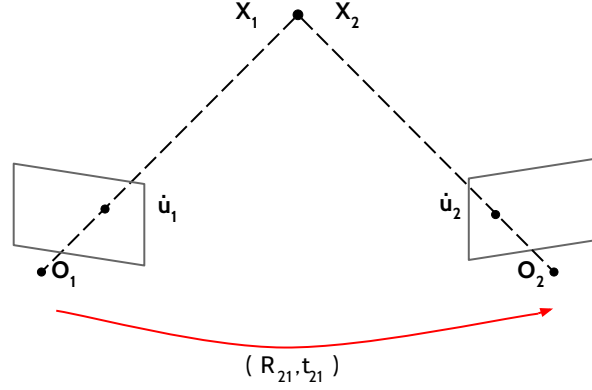
Fonte: Produção do próprio autor.

Na Figura 12, um ponto do corpo da menina da foto está representado pelo ponto tridimensional $\mathbf{X}_1 = (x_1, y_1, z_1)^T$, cuja projeção no plano da imagem é dada por $\mathbf{u}_1 = (u_1, v_1, 1)^T$. Qualquer que seja a posição de \mathbf{X}_1 sobre a linha ℓ_1 ($\mathbf{X}'_1, \mathbf{X}''_1$ e \mathbf{X}'''_1), sua projeção continuará sendo a mesma. Matematicamente, tem-se que para diferentes valores de z_1 haverá um \mathbf{X}_1 que obedeça à equação de reta

$$\mathbf{X}_1(z_1) = z_1 K_1^{-1} \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix}. \quad (2.16)$$

Nota-se que tendo somente a Equação 2.16 não é possível descobrir z_1 , visto que não se tem nenhum elemento do ponto \mathbf{X}_1 , havendo, portanto, infinitas soluções. Entretanto, se houver outra imagem da mesma cena onde se conhece o ponto correspondente do pixel \mathbf{u}_1 , pode-se triangular a informação e reaver a profundidade z_1 , como mostra a Figura 13. Essa segunda foto pode ser fornecida por qualquer câmera posicionada em um local diferente da primeira. Um sistema de duas câmeras é chamado de **sistema estéreo**, entretanto, ao abstrair essa definição para imagens, tem-se que qualquer par de imagens que possua uma transformação levando pixels de uma para a outra pode ser chamado de par estéreo, ou sistema estéreo.

Figura 13 – Triangulação do ponto \mathbf{X}_1 com a adição de uma segunda imagem. (R_{21}, t_{21}) representa a transformação de \mathbf{X}_1 no sistema de coordenadas da câmera 1 para \mathbf{X}_2 no sistema da câmera 2.



Fonte: Produção do próprio autor.

Para se triangular a informação, deve-se encontrar a expressão que relaciona os pixels correspondentes das duas imagens. Portanto, primeiramente, transforma-se o pixel $\dot{\mathbf{u}}_1$ para unidades métricas com

$$\mathbf{X}_1 = z_1 K_1^{-1} \dot{\mathbf{u}}_1, \quad (2.17)$$

considerando z_1 como a variável que se deseja encontrar.

Posteriormente, para encontrar a projeção do ponto \mathbf{X}_1 no plano da imagem da câmera 2, deve-se transformá-lo para o sistema de coordenadas da câmera 2

$$\mathbf{X}_2 = R_{21} \mathbf{X}_1 + t_{21}, \quad (2.18)$$

e, então, projetá-lo no plano da imagem da câmera 2

$$z_2 \dot{\mathbf{u}}_2 = K_2 \mathbf{X}_2. \quad (2.19)$$

Após a projeção, o resultado estará sobre o efeito do fator de escala z_2 , que deve ser retirado dividindo-se o próprio vetor resultado $z_2 \dot{\mathbf{u}}_2$ pelo seu último elemento. Assim, neste caso

$$\begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} = \frac{1}{z_2} \begin{pmatrix} z_2 u_2 \\ z_2 v_2 \\ z_2 \end{pmatrix} = \frac{1}{z_2} z_2 \dot{\mathbf{u}}_2. \quad (2.20)$$

Todo o processo citado pode ser representado pela expressão

$$z_2 \mathbf{\hat{u}}_2 = K_2(R_{21}(z_1 K_1^{-1} \mathbf{\hat{u}}_1) + t_{21}). \quad (2.21)$$

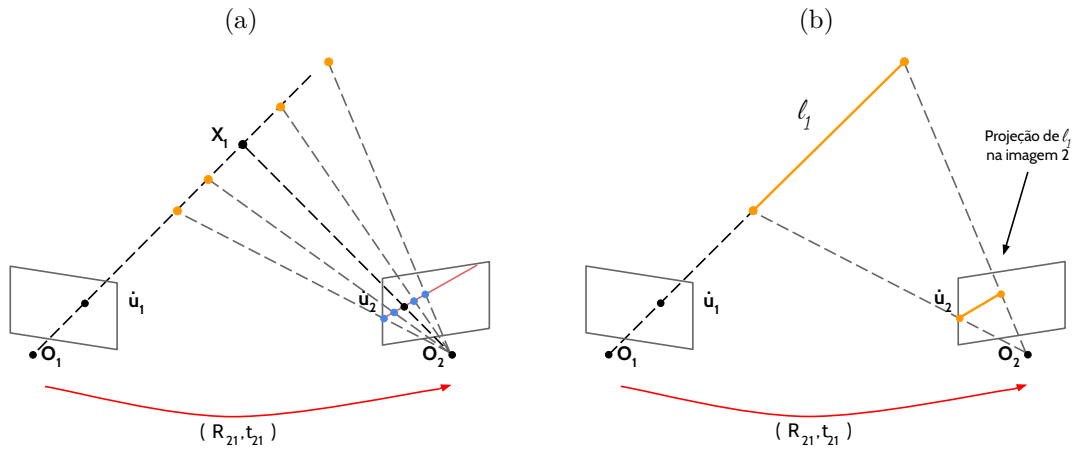
Assumido conhecimento do pixel $\mathbf{\hat{u}}_2$ da imagem 2 correspondente a $\mathbf{\hat{u}}_1$, tem-se que z_1 é o valor que valida a Equação 2.21.

Claro que com uma foto tirada de outro ângulo, é possível identificar que a menina não possui uma saia de flor. Entretanto, ao obter a profundidade da cena, o robô pode se localizar, interagir com o meio e, inclusive, descobrir a proporção entre a moça e a flor.

2.3 Geometria epipolar

A Figura 14 apresenta os pixels correspondentes a $\mathbf{\hat{u}}_1$ na imagem 2 para diferentes valores de z_1 .

Figura 14 – (a) pixels correspondentes a $\mathbf{\hat{u}}_1$ na imagem 2 para diferentes valores de z_1 . (b) A projeção da linha de projeção ℓ_1 no plano da imagem da câmera 2 recebe o nome de linha epipolar.



Fonte: Produção do próprio autor.

A localização do pixel correspondente a $\mathbf{\hat{u}}_1$ na imagem 2 varia segundo a expressão

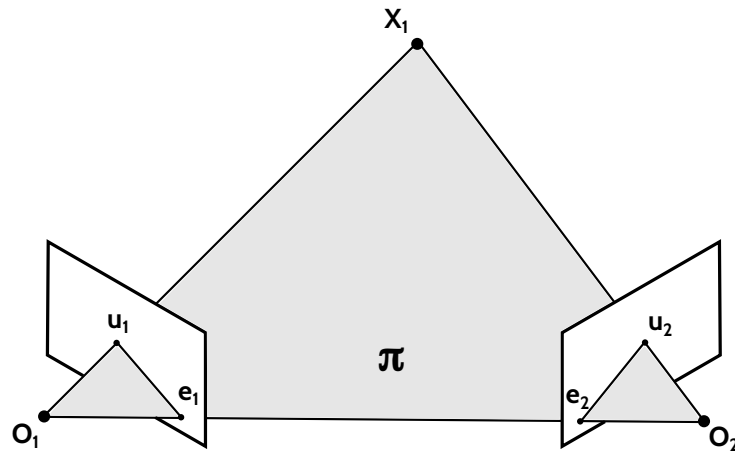
$$z_2 \mathbf{\hat{u}}_2 = K_2(R_{21}(z_1 K_1^{-1} \mathbf{\hat{u}}_1) + t_{21}), \quad (2.22)$$

como apresentado na Seção 2.2.

A Equação 2.22 descreve uma reta, onde para cada valor de z_1 há um pixel $\mathbf{\hat{u}}_2$. Essa reta corresponde à projeção da linha ℓ_1 no plano da imagem da câmera 2, vide Figura 14. Ela recebe o nome de **linha epipolar**.

A Figura 15 ilustra geometricamente a idéia para ambas as câmeras.

Figura 15 – O ponto \mathbf{e}_1 , chamado de **epipolo**, é a projeção do centro óptico da câmera 2 (\mathbf{O}_2) no plano da imagem da câmera 1. O epipolo \mathbf{e}_2 é encontrado da mesma maneira. O plano π é formado pelos pontos \mathbf{O}_1 , \mathbf{O}_2 e \mathbf{X} , onde a linha que liga os centros ópticos das câmeras é chamada de **baseline**. Sua interseção com os planos das imagens formam as linhas epipolares.



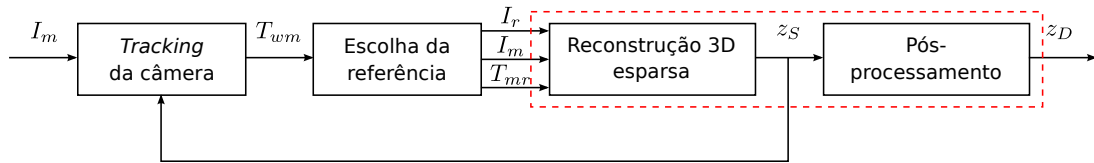
Fonte: Produção do próprio autor.

A geometria que reúne essas relações de visão computacional para um sistema estéreo recebe o nome de **geometria epipolar**. Sua importância para as aplicações de visão está principalmente nas restrições geométricas, que como consequência, limitam o espaço de busca pelo pixel correspondente para uma linha, não sendo necessária uma busca exaustiva por toda a imagem.

3 Método Proposto

O processo, descrito no Capítulo 2, que recupera a informação tridimensional do meio recebe o nome de reconstrução 3D. Esse processo necessita de pelo menos um par de imagens e dos parâmetros intrínsecos e extrínsecos das câmeras conhecidos para se estimar a profundidade da cena, seguindo o modelo apresentado na Figura 16.

Figura 16 – Modelo completo de reconstrução 3D com *tracking* da câmera para extração de parâmetros da câmera. O escopo do trabalho está destacado pelo tracejado vermelho.



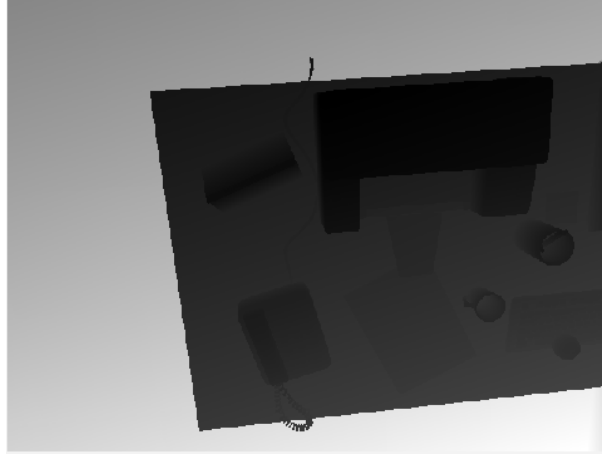
Fonte: Produção do próprio autor.

No modelo completo de reconstrução 3D, mostrado na Figura 16, o sistema captura uma imagem I_m por meio da câmera. A partir de imagens previamente obtidas $(I_r, I_{m-1}, \dots, I_1)$, encontra-se os parâmetros intrínsecos da câmera, e extrínsecos T_{wm} , que relacionam a posição atual da câmera com o referencial global. Uma imagem é escolhida como referência I_r e, com um par de imagens (I_r, I_m) e a relação T_{mr} entre os centros ópticos de I_r e I_m , é possível estimar a profundidade z_S de cada pixel de I_r . Esse resultado de profundidade pode ser usado para melhorar a estimação dos parâmetros da câmera, e pode ser refinado em um pós-processamento, sendo possível encontrar uma nuvem de pontos mais precisa e densa z_D .

A profundidade real da cena (*ground truth*) é normalmente representada por meio de uma imagem em escala de cinza, denominada mapa de profundidade. Quanto mais próximo da câmera, mais escura é a cor, como pode ser visto na Figura 17. Essa também será a representação adotada para apresentar os resultados deste trabalho.

Note que, o foco deste trabalho está apenas na etapa de reconstrução. A estimativa dos parâmetros intrínsecos e extrínsecos para a realização do *tracking* da câmera por si só representa um problema matematicamente complexo, portanto, não será considerada como parte do escopo deste trabalho, sendo tais parâmetros tomados como já conhecidos. Além da reconstrução, também serão tratados os casos de regiões que apresentam pouca textura em um pós-processamento, para se atingir uma nuvem de pontos mais densa. O escopo do trabalho está destacado pelo tracejado vermelho na Figura 16.

Neste capítulo, antes de se mostrar a abordagem adotada, será discutido o DTAM,

Figura 17 – Mapa de profundidade do *ground truth*.

Fonte: Handa et al. (2012).

método escolhido para comparação. O DTAM é um método muito citado na literatura (CONCHA; CIVERA, 2015; CONCHA et al., 2015; ??; ??) por realizar, em tempo real, a reconstrução 3D da cena e a calibração da câmera, apresentando um bom compromisso entre velocidade e qualidade. Além disso, seus resultados são bons o suficiente para aplicações em realidade aumentada (??). Todas essas características motivaram essa escolha.

Em seguida, será apresentada uma visão geral da abordagem adotada para a solução do problema e, posteriormente, explicada cada etapa do processo.

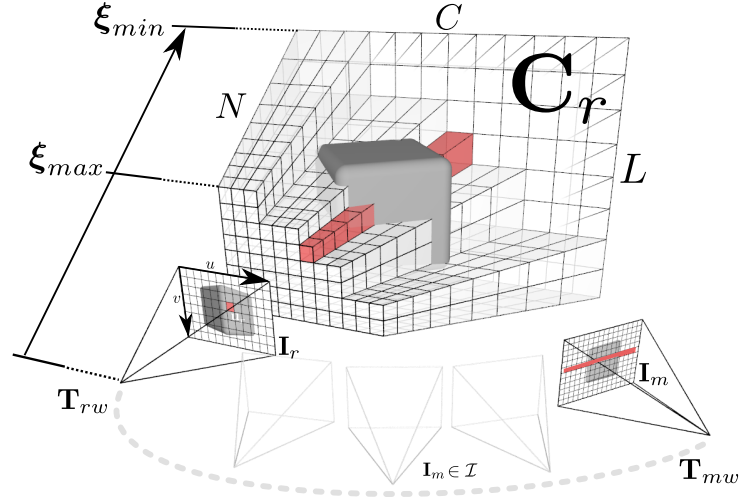
3.1 Dense Tracking And Mapping (DTAM)

O núcleo de reconstrução 3D do DTAM é formado basicamente por duas etapas: a estimativa densa discreta da profundidade dos pixels (estimativa grosseira) e a estimativa densa contínua por meio de um processo iterativo de minimização (mais precisa).

Na primeira etapa, as correspondências dos pixels de uma imagem de referência I_r são buscadas em um conjunto \mathcal{I} de M imagens I_m , como mostrado na Figura 18. A cena é representada por um conjunto de pixels tridimensionais, também conhecidos como *voxels*. Cada *voxel* está ligado a um possível ponto tridimensional dado pelo pixel de I_r e um dos valores de profundidade possíveis dentro da faixa discreta $[z_{min}, z_{max}] = [\frac{1}{\xi_{max}}, \frac{1}{\xi_{min}}]$.

A linha epipolar é definida por essa faixa discreta de N níveis de profundidade ao invés de uma varredura de pixel em pixel na imagem. Então, a discretização é feita, no domínio da imagem, de forma linearmente espaçada em termos de pixels para que a linha epipolar seja varrida uniformemente, à medida que se muda entre os N níveis de profundidade. Por exemplo, dado que para um pixel u_r a linha epipolar deva ser percorrida de $u_{m1} = 1$ a $u_{m2} = 79$, e que $N = 40$ níveis, os pixels percorridos serão $u_m = 1, 3, 5, \dots, 79$.

Figura 18 – Representação da cena por um conjunto de *voxels* para a etapa de reconstrução discreta do DTAM. São utilizadas uma imagem de referência I_r e um conjunto \mathcal{I} de M imagens I_m para tal.



Fonte: Newcombe, Lovegrove e Davison (2011) (adaptado pelo autor).

E por isso, o DTAM trabalha com o inverso da profundidade $\xi = \frac{1}{z}$ em todo o processo. O resultado final z é calculado ao se inverter ξ . Essa forma de discretização usada no DTAM, também é aplicada nesta dissertação.

A métrica utilizada pelo DTAM é a soma do absoluto das diferenças (SAD) pixel a pixel. A explicação dessa escolha está na complexidade baixa do SAD pixel a pixel e de que o número de reconstruções feitas pelo DTAM é grande, o que exige uma abordagem com um custo computacional menor. A função para encontrar a melhor correspondência dos pixels de I_r no conjunto \mathcal{I} é dada por

$$\xi(\mathbf{u}) = \underset{\xi}{\operatorname{argmin}} C_r(\xi, \mathbf{u}), \quad (3.1)$$

onde

$$C_r(\xi, \mathbf{u}) = \frac{1}{M} \sum_{I_m \in \mathcal{I}} |I_r(\mathbf{u}) - I_m(KT_{mr}K^{-1}\frac{1}{\xi}\mathbf{u})|, \quad (3.2)$$

sendo $C_r(\xi, \mathbf{u})$ a média das métricas nas M imagens, caracterizando um filtro passa baixas, que tira as diferenças bruscas entre as diversas reconstruções.

Como pode ser visto da Figura 18, a função de custo $C_r(\xi, \mathbf{u})$ é representada por uma matriz tridimensional de mesma quantidade de linhas e colunas da imagem I_r ($L \times C$) e N canais, sendo que cada canal representa um valor possível de ξ . O resultado final dessa etapa $\xi(\mathbf{u})$ é uma matriz bidimensional com mesmas dimensões que I_r e apresenta os valores de ξ que minimizam $C_r(\xi, \mathbf{u})$ para cada pixel.

Na segunda etapa, se minimiza uma função de energia

$$E_\xi = \int \{R(\xi(\dot{\mathbf{u}}), \dot{\mathbf{u}}) + C_r(\xi(\dot{\mathbf{u}}), \dot{\mathbf{u}})\} d\dot{\mathbf{u}}, \quad (3.3)$$

formada pela função de custo $C_r(\xi(\dot{\mathbf{u}}), \dot{\mathbf{u}})$ e um termo regularizador $R(\xi(\dot{\mathbf{u}}), \dot{\mathbf{u}})$. Esse termo adiciona à função de energia a consideração de que a profundidade do mundo varia suavemente, salvo os limites dos objetos, sendo calculado como

$$R(\xi(\dot{\mathbf{u}}), \dot{\mathbf{u}}) = g(\dot{\mathbf{u}}) \|\nabla \xi(\dot{\mathbf{u}})\|_\epsilon, \quad (3.4)$$

onde

$$\|x\|_\epsilon = \begin{cases} \frac{\|x\|_2^2}{2\epsilon} & \text{se } \|x\|_2 \leq \epsilon \\ \|x\|_1 - \frac{\epsilon}{2} & \text{caso contrário} \end{cases}. \quad (3.5)$$

Assim como mostrado em (NEWCOMBE; LOVEGROVE; DAVISON, 2011), ϵ é uma constante de pequeno valor, na ordem de grandeza de $\approx 1, 0e^{-4}$ e $g(\dot{\mathbf{u}})$ é um peso aplicado ao gradiente do inverso da profundidade resultante $\xi(\dot{\mathbf{u}})$. Seus valores são baixos nas bordas de I_r , dado que as descontinuidades geralmente coincidem com elas.

$\xi(\dot{\mathbf{u}})$ é alterado iterativamente em um processo de otimização podendo assumir valores discretos dentro da faixa definida. É possível aumentar o número de níveis discretos de profundidade N para melhorar a precisão dos resultados. Para obter valores contínuos, o DTAM, a cada iteração, realiza um processo parecido com o da aproximação local por uma parábola visto na Seção 3.3.3.

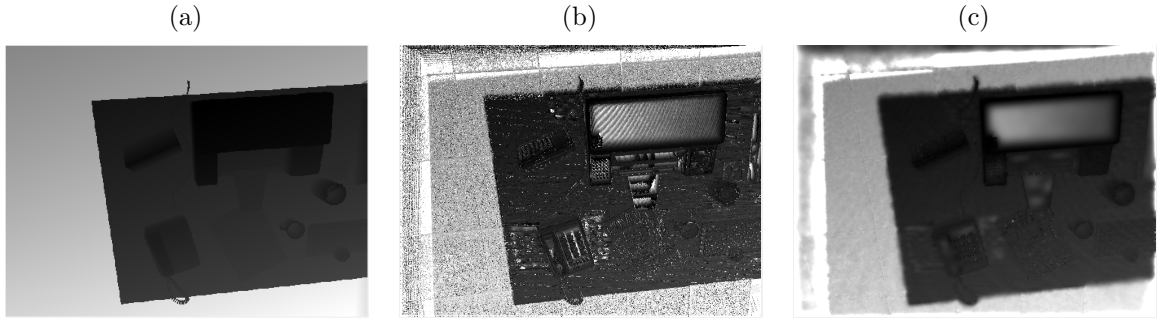
A Figura 19 apresenta o resultado da reconstrução discreta e contínua do DTAM. Pode-se notar erros altos de estimativa sobre diversas regiões da impressora e do telefone (áreas claras que deveriam ser escuras, vide Figura 19a). Por não possuírem muita textura, essas áreas não são bem reconstruídas e, quando muito grosseira, essa estimativa incorreta acaba sendo propagada para o resultado final do DTAM, mesmo após o processo iterativo (Figura 19c).

A Figura 20 apresenta a nuvem de pontos dos resultados apresentados na Figura 19. Como pode-se perceber na Figura 20a, mesmo com a média das funções de métrica, comparar somente a intensidade de um pixel ao invés de utilizar métodos em janela leva a muito erros, deixando o resultado bem ruidoso. Já na Figura 20b o resultado da reconstrução atingiu uma superfície mais suave após o processo iterativo, entretanto, continua a apresentar altos erros nas regiões com pouca textura.

3.2 Visão geral da abordagem proposta

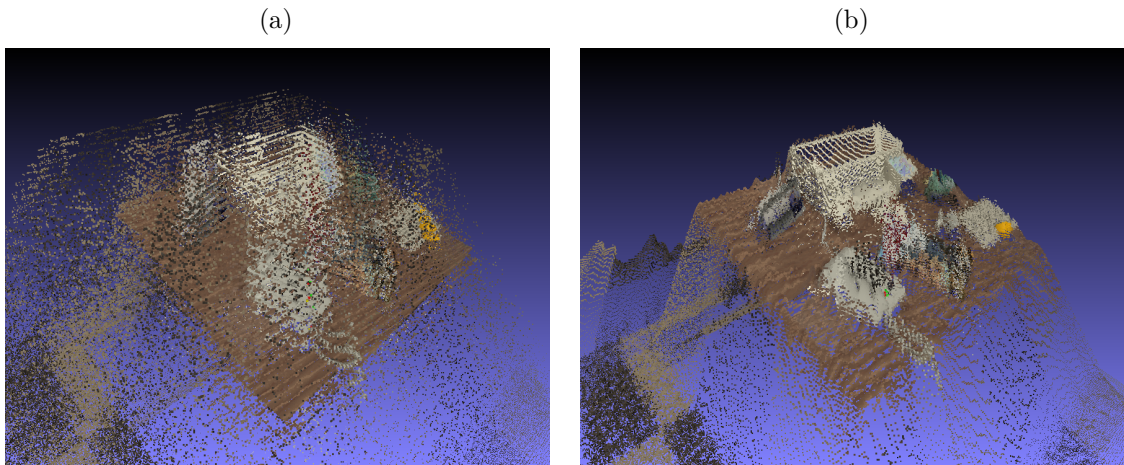
O sistema proposto divide-se em dois processos: reconstrução 3D esparsa (Seção 3.3) e tratamento de regiões homogêneas (Seção 3.4). A Figura 21 apresenta resumidamente o

Figura 19 – (a) *Ground truth* e mapas de profundidade da estimativa discreta (b) e contínua (c) do sistema DTAM. Foram utilizadas 30 imagens com $N = 32$ níveis de profundidade. O erro médio absoluto em relação ao *ground truth* de (b) foi de 24,27 cm e o de (c) foi de 23,09 cm. O percentual total de pixels reconstruídos em ambas é de 100 % pela reconstrução do DTAM ser de todos os pixels da imagem de referência.



Fonte: Produção do próprio autor.

Figura 20 – Nuvem de pontos do resultado apresentado na Figura 19b (a) e na Figura 19c (b).



Fonte: Produção do próprio autor.

esquemático do algoritmo do processo de reconstrução esparsa.

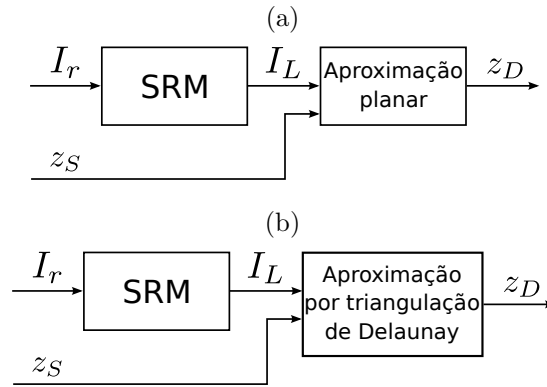
No método aqui proposto, é utilizado um conjunto \mathcal{I} de 10 imagens I_m ($m = 1, \dots, 10$) que possuem certa sobreposição com uma imagem de referência I_r . Todas as imagens utilizadas pertencem ao espaço de cor RGB e as matrizes que levam do sistema de coordenadas da câmera r para as coordenadas de cada câmera m estão representadas na Figura 21 como T_{mr} . Um exemplo de imagem de referência que foi utilizada neste trabalho está apresentada na Figura 22.

Cada par (I_r, I_m) é retificado gerando um par (I_{rR}, I_{mR}) . Dessa forma, os pixels de uma imagem podem ser encontrados na mesma linha da outra, ajudando na etapa de

Dados os correspondentes, é possível calcular a profundidade z_r dos pixels de I_r . Os 10 resultados $z_{r[1-10]}$ estão associados à profundidade z_r referente a cada par. Esses resultados são filtrados por duas etapas, filtro temporal e filtro espacial (Seção 3.3.5), onde são retirados pixels com possíveis erros de correspondência, dando origem a z_S .

Os pixels reconstruídos, após as etapas de filtragem, representam uma nuvem de pontos esparsa. Para que a reconstrução 3D torne-se densa, duas abordagens foram analisadas, como pode ser visto na Figura 23.

Figura 23 – Esquemático resumido dos algoritmos de tratamento de regiões homogêneas. (a) algoritmo que aproxima cada região segmentada por um plano. (b) algoritmo que aproxima uma superfície por meio da triangulação de Delaunay dentro de cada região.



Fonte: Produção do próprio autor.

Nos dois casos, a imagem I_r é segmentada em regiões homogêneas de cor similar utilizando a técnica *Statistical Region Merging* (SRM) proposta por (NOCK; NIELSEN, 2004), que retorna uma matriz I_L contendo os rótulos (*labels*) de cada região. I_L é uma matriz $2D$ de mesmo tamanho que I_r . No final de ambos processos, o resultado de estimativa densa de profundidade está representado por z_D .

Inicialmente (primeira abordagem), analisou-se a consideração de que cada região de I_L representa um plano. Dessa forma, os pontos reconstruídos e não descartados pela etapa de filtragem podem ser utilizados para estimar o vetor normal ao plano correspondente a cada região (\vec{n}) e distância do plano à câmera (d). Para retirar possíveis *outliers*, nesse processo, utilizou-se o algoritmo RANSAC (FISCHLER; BOLLES, 1981).

Posteriormente, resolveu-se investigar os casos onde cada região de I_L corresponde a uma área plana ou, pelo menos, sem descontinuidades (segunda abordagem). Para esses casos, a aproximação de planos pode levar a resultados equivocados. Por isso, utilizou-se a triangulação de Delaunay (DELAUNAY, 1934) nos pontos reconstruídos de cada região, para que a aproximação de planos se torne local em triângulos formados a cada 3 pixels reconstruídos. Um filtro por histograma da profundidade desses pixels foi utilizado para

caso alguma região apresente descontinuidades.

3.3 Reconstrução 3D esparsa

Na etapa de reconstrução esparsa, os pixels de I_r ($\hat{\mathbf{u}}_r$) são procurados sobre a linha epipolar em cada imagem I_m para que se possa realizar a triangulação de informação e se estimar as 10 hipóteses de profundidade. Esse processo é chamado na literatura de **matching** (HARTLEY; ZISSERMAN, 2003). Nele, uma métrica quantifica a similaridade entre os pixels de I_m ($\hat{\mathbf{u}}_m$) e de I_r ($\hat{\mathbf{u}}_r$). Posteriormente, escolhe-se a correspondência de cada ponto com alguma estratégia de decisão como a *winner-takes-all*¹. Esse procedimento será abordado com detalhes nessa seção, assim como a filtragem de possíveis correspondências ruins (*outliers*).

3.3.1 Matching

Apesar de existirem muitas métricas diferentes na literatura, as métricas SAD (soma do absoluto das diferenças), SSD (soma do quadrado das diferenças) e NCC (correlação cruzada normalizada) são amplamente utilizadas em reconstrução 3D (HIRSCHMULLER; SCHARSTEIN, 2009; FURUKAWA; HERNÁNDEZ et al., 2015; CHANG et al., 2013). Esses métodos comparam a intensidade de pixels no interior de uma janela centrada nos pixels de I_m e I_r , e por isso são classificados como **métricas baseadas em janela**. Uma alternativa é a comparação exclusiva da intensidade do pixel com o possível correspondente, que será chamada neste trabalho de **métrica pixel a pixel**². Quanto menos comparações, menos complexo e mais rápido será o método. Um exemplo com imagens usadas neste trabalho pode ser conferido na Figura 24.

No exemplo da Figura 24 foi utilizado como métrica o SAD para o caso de um pixel

$$SAD = |I_r(v_r, u_r) - I_m(v_m, u_m)|, \quad (3.6)$$

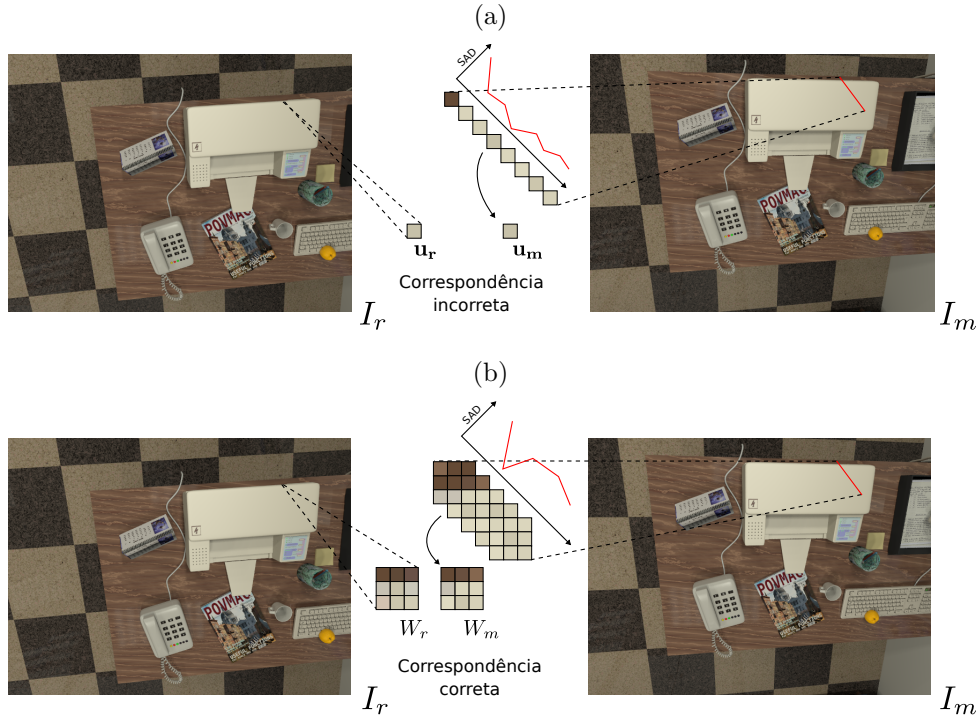
e para o caso de uma janela W_r em torno do pixel $\hat{\mathbf{u}}_r$

$$SAD = \sum_{(j,k) \in W_r} |I_r(k, j) - I_m(k + (v_m - v_r), j + (u_m - u_r))|. \quad (3.7)$$

¹ *winner-takes-all* é uma estratégia de decisão onde se escolhe como correspondência o ponto que apresentar a melhor métrica.

² O SSD e o SAD são métricas que podem ser baseadas em janela ou em pixel a pixel.

Figura 24 – (a) processo de correspondência utilizando SAD para um pixel. (b) processo de correspondência utilizando SAD com janela de 3×3 pixels.



Fonte: Produção do próprio autor.

$I_m(k + (v_m - v_r), j + (u_m - u_r)) \forall (j, k) \in W_r$ representa a janela W_m em torno do pixel \hat{u}_m . A Equação 3.7 pode ser apresentada de uma forma mais simples como

$$SAD(W_r, W_m) = \sum |W_r - W_m|. \quad (3.8)$$

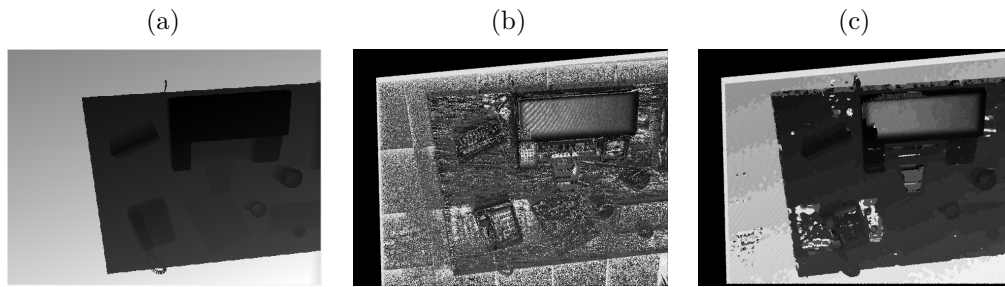
Como mostram as Equações 3.6 e 3.7, a diferença no processo entre o caso de um pixel para uma janela em torno dele é que, no caso da janela, soma-se as n^2 diferenças absolutas (janela $n \times n$) calculadas. Para ambos casos, quanto menor o valor da métrica, menor será a diferença entre os pixels, indicando uma maior similaridade. Entretanto, usar somente um pixel não levará em consideração características da vizinhança. E, como pode-se notar na Figura 24, o que difere \hat{u}_r dos demais pixels da região homogênea que ele se encontra é o fato dele estar na borda dessa região. Não utilizar uma janela em torno de \hat{u}_r significa não utilizar essa informação de borda, ou seja, ele será parecido com qualquer outro pixel do interior da região. Dessa forma, aparecem muitos pixels com métrica baixa e semelhante - grandes potenciais correspondentes.

Apesar de ser intuitivo que a diferença entre \hat{u}_r e o correspondente correto deva ser 0, devido à discretização da cena pelos sensores da câmera e a variações de iluminação, os correspondentes entre imagens podem não apresentar mesma intensidade de cor. Isso também acontece nos métodos baseados em janela. Entretanto, ao se utilizar informação

da vizinhança, adiciona-se à métrica muitas vezes características marcantes que servirão de pontos de referência na localização correta do pixel - existem muitos prédios brancos em uma cidade, já um que fica entre um prédio azul, um laranja e um rosa é muito mais difícil de se errar. Traduzindo, utilizar uma janela auxilia na busca pela correspondência de $\hat{\mathbf{u}}_{\mathbf{r}}$ em I_m .

O resultado final da estimação da profundidade para o caso de janelas e de pixel a pixel desse exemplo pode ser visto na Figura 25, onde pode-se notar a quantidade de ruído no caso pixel a pixel em relação ao caso de janelas. Isso também se reflete no erro médio absoluto das duas abordagens, sendo o advindo do método que utiliza somente um pixel quase o triplo do outro.

Figura 25 – (a) *Ground truth* e mapas de profundidade da estimação utilizando SAD pixel a pixel (b) e janelas de 7×7 pixels (c). O erro médio absoluto em relação ao *ground truth* de (b) foi de 81,00 cm e o de (c) foi de 28,55 cm. A área preta que compreende o canto superior esquerdo representa os pixels que não foram reconstruídos por serem pontos da cena que não estavam na imagem 2.



Fonte: Produção do próprio autor.

Além dos métodos de janela e pixel a pixel, pode-se utilizar técnicas de descrição e correspondência, como é o caso do SURF (BAY; TUYTELAARS; GOOL, 2006) e do SIFT (LOWE, 1999). Essas técnicas encontram regiões com características marcantes e retornam o pixel central dessas regiões e um descritor que quantifica essas características. Depois, encontram-se, entre imagens, pontos correspondentes com a comparação entre seus descritores. A vantagem desses métodos está em serem invariantes à rotação e à escala. Entretanto, essas técnicas não retornam muitos pontos das imagens, o que não é ideal para uma reconstrução 3D densa.

No processo de reconstrução deste trabalho, as métricas SAD, SSD e NCC serão analisadas no Capítulo 4 com o intuito de descobrir qual a melhor a se usar ante o objetivo de ter rapidez no sistema, mas ao mesmo tempo não perder em qualidade de reconstrução final. As janelas usadas terão 7×7 pixels e três canais de cor, RGB. O SSD é calculado

como

$$SSD(W_r, W_m) = \sum (W_r - W_m)^2, \quad (3.9)$$

onde a diferença entre cada pixel de W_r com de W_m é elevada ao quadrado. A diferença quadrática de todos os pixels da janela são somados para o cálculo do SSD.

Já no caso do NCC, de maneira similiar a (GOESELE; CURLESS; SEITZ, 2006) e (ALCANTARILLA; BEALL; DELLAERT, 2013), neste trabalho, utiliza-se uma versão do NCC l -dimensional para adaptação ao espaço de cor RGB com normalização por canal de cor

$$NCC(W_r, W_m) = \frac{\sum (W_r - \bar{W}_r) \cdot (W_m - \bar{W}_m)}{\sqrt{\sum (W_r - \bar{W}_r)^2 \cdot \sum (W_m - \bar{W}_m)^2}}. \quad (3.10)$$

\bar{W} é a média da intensidade dos pixels da janela W para cada canal de cor RGB. O NCC retorna um número no intervalo de $[-1, 1]$, sendo que 1 indica máxima correlação. Para diminuir o tempo de processamento, o cálculo do NCC foi feito baseado na implementação proposta por Lewis (1995), chamada de *Fast Normalized Cross-Correlation* (FNCC), como pode ser visto na Equação 3.11. Além disso, também se utiliza imagem integral para o cálculo rápido das médias das janelas (CROW, 1984).

$$FNCC(W_r, W_m) = \frac{\sum W_r \cdot (W_m - \bar{W}_m)}{\sqrt{\sum (W_r - \bar{W}_r)^2 \cdot \left(\sum W_m^2 - \frac{1}{n^2} \sum_{l=0}^2 \left(\sum_{(j,k) \in W_m} W_m(k, j, l) \right)^2 \right)}} \quad (3.11)$$

Na Equação 3.11, $W(k, j, l)$ indica o valor de intensidade do canal de cor l do pixel (j, k) da janela W de tamanho $n \times n$ pixels, onde $l = \{0, 1, 2\}$ representa os canais RGB, respectivamente. $\sum W$ retorna a soma da intensidade de todos os pixels da janela W ($\sum_{(j,k) \in W} W$) e de cada canal l . Ao se utilizar imagem integral, os termos $\sum W^2$ e $\sum W$ podem ser calculados rapidamente.

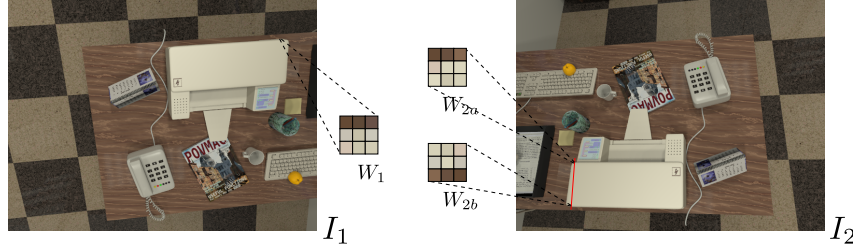
3.3.2 Retificação epipolar

Assim como mostrado no Capítulo 2, devido à restrição epipolar, um pixel \mathbf{u}_1 na imagem 1 pode ser encontrado na imagem 2 sobre a linha epipolar definida pela equação de reta

$$z_2 \dot{\mathbf{u}}_2 = K_2(R_{21}(z_1 K_1^{-1} \dot{\mathbf{u}}_1) + t_{21}). \quad (3.12)$$

Essa restrição reduz o universo de busca de uma imagem para uma linha na etapa de *matching*, o que simplifica bastante o problema. A Figura 26 apresenta essa ideia, onde o pixel \mathbf{u}_1 da imagem I_1 é procurado na imagem I_2 sobre a linha epipolar.

Figura 26 – Exemplo de duas possíveis correspondências de W_1 em I_2 : W_{2a} e W_{2b} . Métrica NCC de W_{2a} e W_{2b} em relação a W_1 é de 0,92 e 0,37, respectivamente.



Fonte: Produção do próprio autor.

Na Figura 26, três janelas estão destacadas: a janela em torno de \mathbf{u}_1 (W_1) e duas janelas (W_{2a} e W_{2b}) em torno de possíveis correspondências sobre a linha epipolar em I_2 , sendo W_{2b} a janela que apresenta o resultado correto. Entretanto, como pode-se notar, a janela que apresentou maior correlação com W_1 pelo método NCC foi W_{2a} . Isso se deve ao fato de que a imagem I_2 possui uma rotação em relação a I_1 suficiente para que a posição dos pixels de W_{2b} não coincida com os de W_1 , contribuindo para que outro ponto possa apresentar uma correlação maior que W_{2b} (como aconteceu com W_{2a}).

Uma solução para esse problema é a **retificação**. Ela transforma as imagens de forma que suas linhas epipolares fiquem paralelas e horizontais. Assim, um pixel \mathbf{u}_{1R} da imagem retificada I_{1R} tem a mesma coordenada vertical v que seu correspondente \mathbf{u}_{2R} da imagem I_{2R} também retificada, preservando, dessa forma, a orientação das janelas. A Figura 27 apresenta um exemplo do resultado da retificação em um par de imagens, onde pode-se notar essa propriedade.

Nesse processo de retificação encontram-se duas transformações lineares, H_1 e H_2 , que aplicadas às imagens I_1 e I_2 levem seus respectivos epipolos \mathbf{e}_1 e \mathbf{e}_2 para o infinito no eixo horizontal, assim como mostrado na Figura 28, levando por consequência às linhas epipolares a ficarem horizontais.

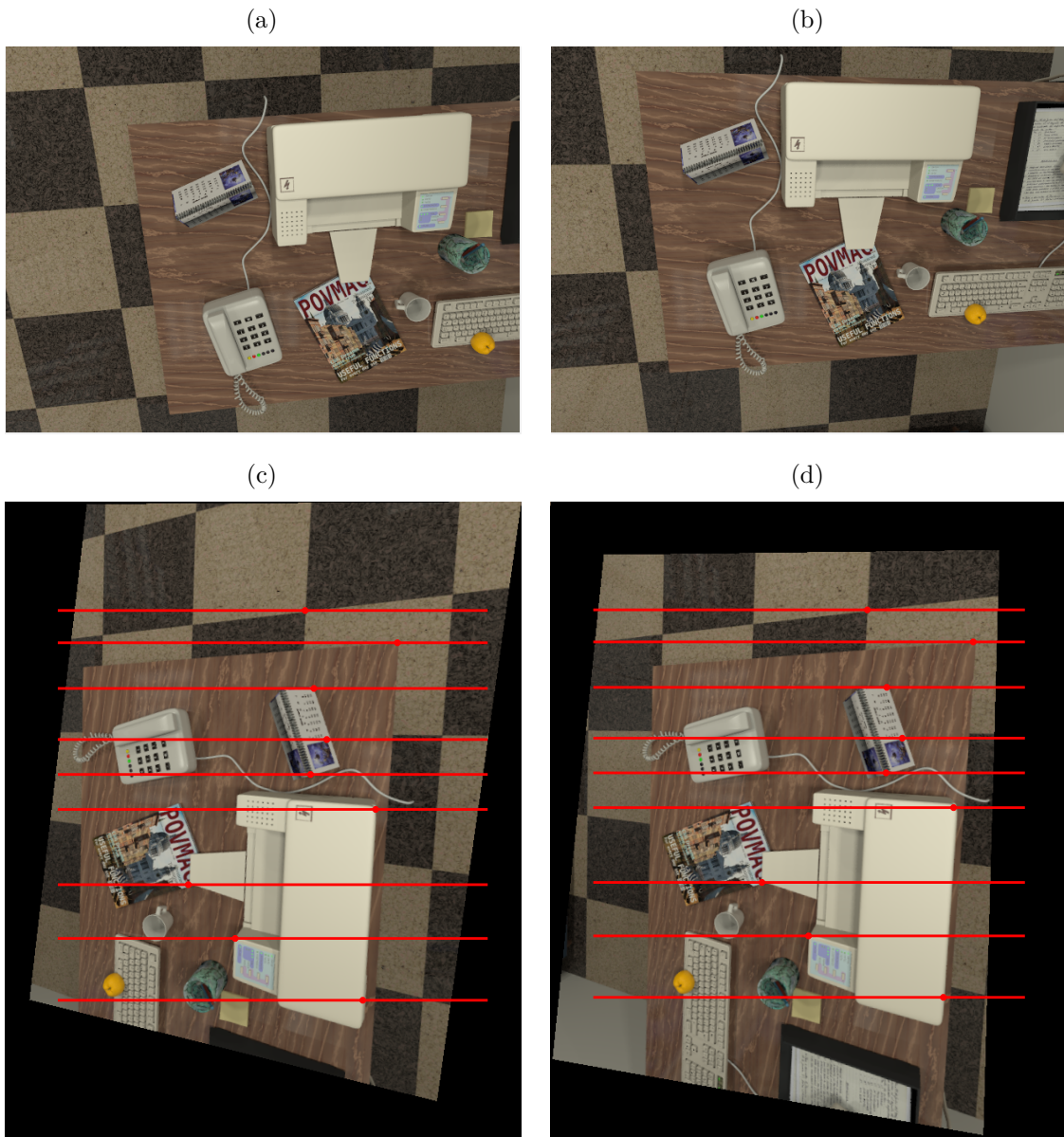
Em outras palavras, buscam-se H_1 e $H_2 \in \mathbb{R}^{3 \times 3}$ que satisfaçam

$$H_1 \mathbf{e}_1 \sim (1, 0, 0)^T, \quad H_2 \mathbf{e}_2 \sim (1, 0, 0)^T, \quad (3.13)$$

onde para qualquer par de pixels correspondentes $(\mathbf{u}_1, \mathbf{u}_2)$, após a retificação $(\mathbf{u}_{1R}, \mathbf{u}_{2R})$, suas coordenadas verticais v sejam iguais (MA et al., 2012).

Na literatura existem muitos algoritmos para encontrar transformações que satisfaçam a Equação 3.13 (MONASSE; MOREL; TANG, 2010; HARTLEY, 1999; FUSIELLO;

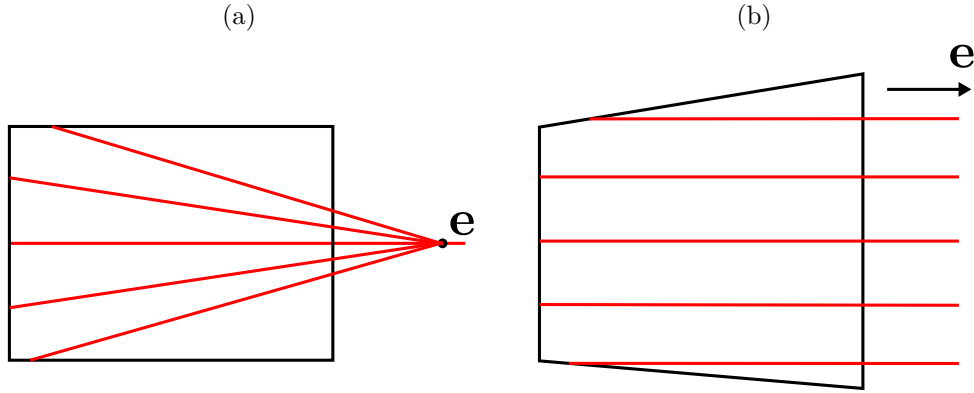
Figura 27 – (a) imagem I_1 . (b) imagem I_2 . (c) imagem retificada I_{1R} . (d) imagem retificada I_{2R} . As linhas epipolares estão representadas em vermelho.



Fonte: Produção do próprio autor.

IRSARA, 2008; MALLON; WHELAN, 2005; KO et al., 2016). Neste trabalho, foi utilizado o método proposto por Fusiello e Irsara (2008), que assume um sistema estéreo calibrado que possua as características apresentadas na Figura 29a. Isto é, assume-se que as câmeras estão em posições em que seus planos de imagem não estejam a frente um do outro, como ilustrado na Figura 29b. Nota-se que para esses casos o deslocamento entre uma câmera e outra é praticamente um movimento para frente (no eixo óptico). Isso traz os epipolos para dentro dos planos da imagem, o que gera um conjunto de linhas epipolares com foco de contração ou expansão.

Figura 28 – (a) plano da imagem com epipolo e e linhas epipolares em vermelho. (b) plano da imagem retificado com linhas epipolares horizontais e epipolo no infinito do eixo horizontal.



Fonte: Gerig (2012) (adaptado pelo autor).

Já no caso da Figura 29a, a componente lateral do movimento entre uma câmera e outra é bem mais significativo. Nesses casos, quando se realiza a retificação, é possível obter as imagens com linhas epipolares horizontais e informação visual válida. No caso da Figura 29b, a retificação gera imagens praticamente perpendiculares ao movimento da câmera e com isso, perde-se a informação visual válida.

O sistema não precisa estar calibrado (HARTLEY; GUPTA, 1993; HEBERT et al., 1995), entretanto, isso permite o uso de técnicas mais simples para retificação.

A ideia é que, ao se retificar as imagens, os centros ópticos das câmeras não sejam alterados, ou seja, H_1 e H_2 devem representar transformações de rotação sem nenhuma translação. Além disso, para que os epipolos sejam levados para o infinito, deve-se definir o novo eixo horizontal como a própria *baseline*. A Figura 30 apresenta o novo sistema de coordenadas que se deseja encontrar.

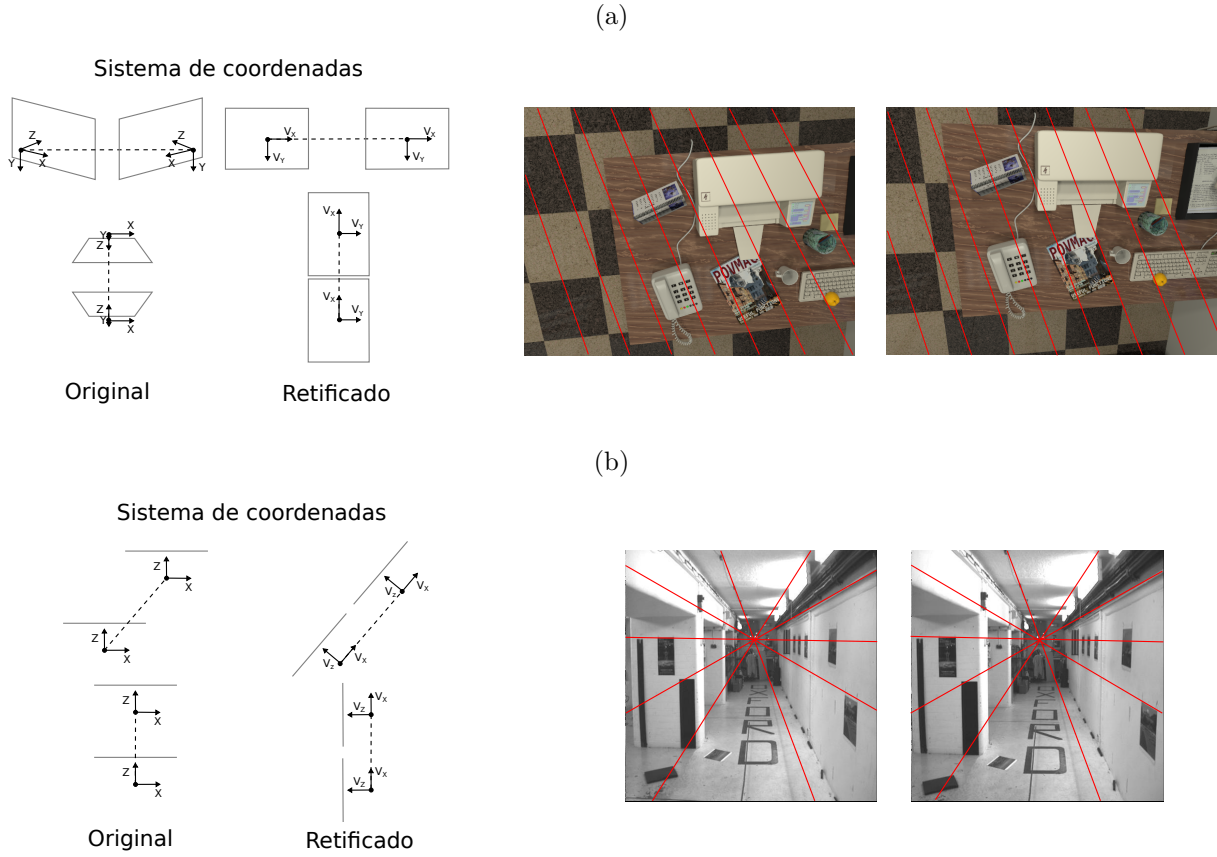
Num caso mais geral, há um sistema de coordenadas global S_w e o sistema das duas câmeras S_1 e S_2 . Nesse caso, a *baseline* é dada pelo vetor que liga os centros ópticos O_1 e O_2 nesse sistema de coordenadas, portanto,

$$O_{1w} = -R_{1w}^T t_{1w}, \quad O_{2w} = -R_{2w}^T t_{2w}, \quad (3.14)$$

$$\vec{b} = O_{2w} - O_{1w}, \quad (3.15)$$

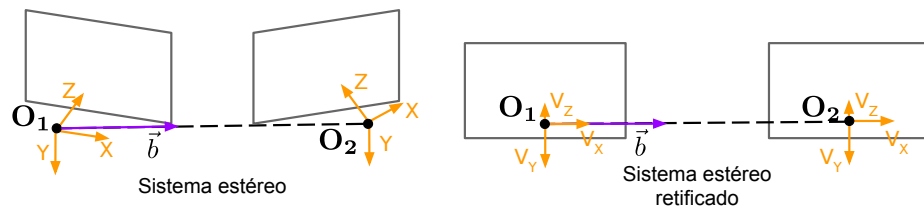
onde R_{1w} e R_{2w} são matrizes de rotação que levam do referencial global para o das câmeras 1 e 2, respectivamente, t_{1w} e t_{2w} suas matrizes de translação. O_{1w} e O_{2w} são os centros ópticos representados no referencial global e \vec{b} a *baseline* ($|\vec{b}| = b$).

Figura 29 – Diferentes arranjos de sistemas compostos por duas câmeras. (a) Disposições onde os epípolos não se encontram dentro da imagem. (b) Disposições onde os epípolos se encontram dentro da imagem. Linhas epipolares estão em vermelho.



Fonte: Produção do próprio autor.

Figura 30 – Os novos eixos X , Y e Z para cada câmera estão apresentados como V_x , V_y e V_z , respectivamente.



Fonte: Produção do próprio autor.

Se a câmera 1 for adotada como referência,

$$\mathbf{O}_1 = (0, 0, 0)^T, \quad \mathbf{O}_2 = -R_{21}^T t_{21}, \quad (3.16)$$

$$\vec{b} = \mathbf{O}_2 - \mathbf{O}_1, \quad (3.17)$$

sendo \mathbf{O}_1 e \mathbf{O}_2 , pontos no sistema de coordenadas da câmera 1.

Dessa forma, o vetor \vec{v}_x normalizado que representa o novo eixo X é dado por

$$\vec{v}_x = \frac{\mathbf{O}_2 - \mathbf{O}_1}{\|\mathbf{O}_2 - \mathbf{O}_1\|_2}. \quad (3.18)$$

O vetor \vec{v}_y normalizado que representa o novo eixo Y deve ser ortogonal ao novo eixo X e ao antigo eixo Z

$$\vec{v}_y = \frac{\vec{z} \times \vec{v}_x}{\|\vec{z} \times \vec{v}_x\|_2}, \quad (3.19)$$

onde \vec{z} é o vetor base do antigo eixo Z .

Já o vetor \vec{v}_z normalizado que representa o novo eixo Z deve ser ortogonal aos novos eixos X e Y

$$\vec{v}_z = \frac{\vec{v}_x \times \vec{v}_y}{\|\vec{v}_x \times \vec{v}_y\|_2}. \quad (3.20)$$

Esse novo sistema de coordenadas define a matriz de rotação R_{Rw} que leva do referencial global para o retificado

$$R_{Rw} = \begin{pmatrix} \vec{v}_x^T \\ \vec{v}_y^T \\ \vec{v}_z^T \end{pmatrix}. \quad (3.21)$$

Resumindo, as imagens 1 e 2 são retificadas aplicando-se a transformação dada na Equação 3.21 em cada pixel para o referencial retificado. Faz-se então

$$z_{1R} \dot{\mathbf{u}}_{1R} = H_1 z_1 \dot{\mathbf{u}}_1, \quad (3.22)$$

e

$$z_{2R} \dot{\mathbf{u}}_{2R} = H_2 z_2 \dot{\mathbf{u}}_2, \quad (3.23)$$

onde

$$H_1 = K_1 R_{Rw} R_{w1} K_1^{-1} \quad (3.24)$$

e

$$H_2 = K_2 R_{Rw} R_{w2} K_2^{-1}, \quad (3.25)$$

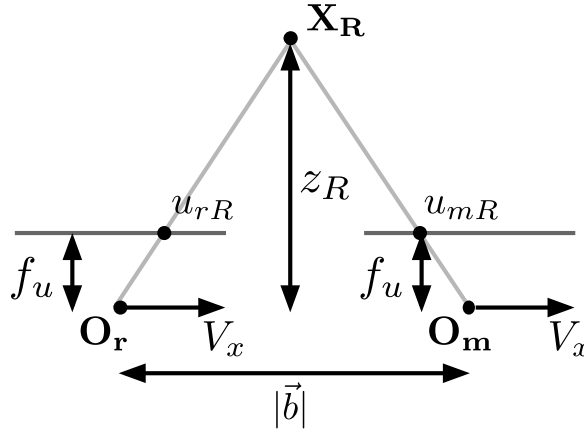
visto que as transformações apenas rotacionam os referenciais em relação aos seus centros ópticos, não havendo translação.

Como no trabalho aqui proposto são utilizadas 10 imagens I_m e uma imagem de referência³ I_r , esse processo se repete 10 vezes, uma para cada par (I_m, I_r) . Os pixels $\dot{\mathbf{u}}_r$ também são transformados para o referencial de I_{rR} por meio da matriz de transformação H_r obtida nesse processo de retificação, conforme Equação 3.26.

$$z_{rR}\dot{\mathbf{u}}_{rR} = K(R_{Rr}R_{rr}K^{-1}z_r\dot{\mathbf{u}}_r) = H_r z_r \dot{\mathbf{u}}_r. \quad (3.26)$$

sendo K a matriz de parâmetros intrínsecos adotada para as câmeras r e m . Após a retificação, os planos da imagem das duas câmeras ficarão como apresentado na Figura 31.

Figura 31 – Sistema de imagens estéreo após a retificação. Planos da imagem paralelos a *baseline*.



Fonte: Produção do próprio autor.

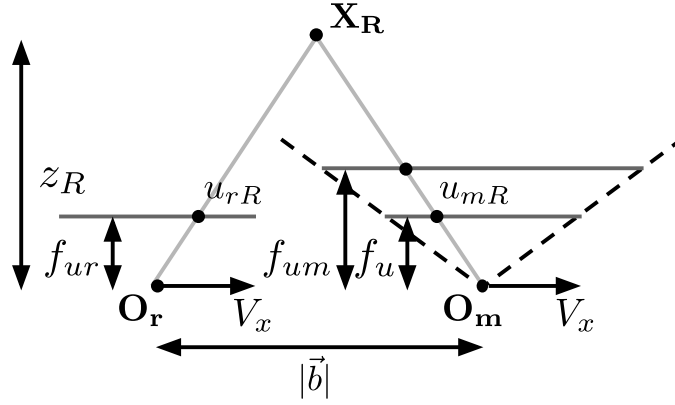
Caso as matrizes de parâmetros intrínsecos das câmeras sejam diferentes, haverá uma matriz K_r e uma K_m e poderá acontecer de os planos da imagem das câmeras retificadas não ficarem no mesmo plano. Matematicamente, isso se reflete no valor de f_u e f_v das duas câmeras não serem iguais. Assim, para levar o plano de I_{mR} para o mesmo que I_{rR} , deve-se encontrar o novo H_m que inclui uma transformação de *zoom*, que aproxime ou afaste o plano da imagem de I_{mR} do seu centro óptico. A Equação 3.27 apresenta a nova matriz H_m .

$$H_m = \begin{pmatrix} \alpha_u & 0 & 0 \\ 0 & \alpha_v & 0 \\ 0 & 0 & 1 \end{pmatrix} H_m, \quad (3.27)$$

onde $\alpha_u = f_{ur}/f_{um}$ e $\alpha_v = f_{vr}/f_{vm}$, ou seja, essa matriz irá escalar a imagem, como mostra a Figura 32.

³ I_r está no sistema de coordenadas de **referência**, ou seja, no sistema adotado como referencial **global** ($Rwr = Rrr = \mathbb{I}_3$).

Figura 32 – Sistema de imagens estéreo após a retificação para $f_{ur} \neq f_{um}$. Ao se aplicar uma transformação de *zoom* (representada pelas linhas tracejadas), ambos planos da imagem se encontram no mesmo plano, com distância dos centros ópticos de f_{ur} .



Fonte: Produção do próprio autor.

Com as imagens retificadas, inicia-se a busca dos correspondentes de $\hat{\mathbf{u}}_{rR}$ sobre as linhas epipolares de I_{mR} , agora horizontais. Como os referenciais das duas câmeras retificadas possuem mesma orientação mas estão transladados de b (*baseline*) no eixo horizontal, o pixel $\hat{\mathbf{u}}_{rR}$ é encontrado na imagem I_{mR} com

$$z_{mR} \begin{pmatrix} u_{mR} \\ v_{mR} \\ 1 \end{pmatrix} = K \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} K^{-1} z_{rR} \begin{pmatrix} u_{rR} \\ v_{rR} \\ 1 \end{pmatrix} - \begin{pmatrix} b \\ 0 \\ 0 \end{pmatrix}, \quad (3.28)$$

$$z_{mR} \begin{pmatrix} u_{mR} \\ v_{mR} \\ 1 \end{pmatrix} = z_{rR} \begin{pmatrix} u_{rR} \\ v_{rR} \\ 1 \end{pmatrix} - K \begin{pmatrix} b \\ 0 \\ 0 \end{pmatrix}. \quad (3.29)$$

Essa relação também pode ser encontrada utilizando igualdade de triângulos na Figura 31. Das duas formas, chega-se em

$$z_{mR} = z_{rR} = z_R, \quad (3.30)$$

$$v_{mR} = v_{rR} = v_R \quad (3.31)$$

e

$$u_{mR} = u_{rR} - \frac{bf_u}{z_R}. \quad (3.32)$$

O correspondente de $\dot{\mathbf{u}}_{\mathbf{r}R}$ é o $\dot{\mathbf{u}}_{\mathbf{m}R}$ que maximiza a similaridade entre as janelas W_{mR} e W_{rR} , logo,

$$\dot{\mathbf{u}}_{\mathbf{m}R} = \underset{\dot{\mathbf{u}}_{\mathbf{m}R}}{\operatorname{argmin}} SAD(W_{mR}, W_{rR}), \quad (3.33)$$

ou

$$\dot{\mathbf{u}}_{\mathbf{m}R} = \underset{\dot{\mathbf{u}}_{\mathbf{m}R}}{\operatorname{argmin}} SSD(W_{mR}, W_{rR}), \quad (3.34)$$

ou

$$\dot{\mathbf{u}}_{\mathbf{m}R} = \underset{\dot{\mathbf{u}}_{\mathbf{m}R}}{\operatorname{argmax}} NCC(W_{mR}, W_{rR}). \quad (3.35)$$

W_{rR} e W_{mR} são janelas de tamanho 7×7 pixels em torno de $\dot{\mathbf{u}}_{\mathbf{r}R}$ em I_{rR} e $\dot{\mathbf{u}}_{\mathbf{m}R}$ em I_{mR} , respectivamente. Os valores que $\dot{\mathbf{u}}_{\mathbf{m}R}$ pode apresentar para cada $\dot{\mathbf{u}}_{\mathbf{r}R}$ obedecem à Equação 3.32 variando-se z_R de um valor mínimo de profundidade estipulado, z_{minR} , até um valor máximo z_{maxR} . $\dot{\mathbf{u}}_{\mathbf{m}R}$ varia de pixel em pixel dentro dessa faixa. Isso reduz ainda mais o universo de busca, dispensando a busca da correspondência por toda a linha epipolar.

Dado o deslocamento $(u_{rR} - u_{mR})$ do pixel nas imagens, também conhecido como **disparidade**, a profundidade z_R pode ser encontrada por meio da Equação 3.32, que com manipulações algébricas pode ser reescrita como

$$z_R = \frac{bf_u}{u_{rR} - u_{mR}}. \quad (3.36)$$

Os valores de z_{minR} e z_{maxR} poderiam ser escolhidos diretamente, entretanto, estes seriam valores dentro do referencial retificado. Ou seja, como a imagem de referência é I_r e não I_{rR} , esses valores de profundidade ainda têm que ser transformados de volta. A própria Equação 3.26 relaciona a profundidade estimada no referencial retificado z_{rR} (z_R) com a profundidade no referencial original z_r como

$$z_{rR} = \gamma z_r, \quad (3.37)$$

sendo γ diferente para cada pixel, dado que $\gamma = h_{r3}\dot{\mathbf{u}}_{\mathbf{r}}$.⁴

Neste trabalho, foram escolhidos os valores mínimo e máximo de profundidade iguais a 10 *cm* e 10 *m*, respectivamente. Desta forma, quando os pixels $\dot{\mathbf{u}}_{\mathbf{r}}$ fossem transformados pela Equação 3.26, o fator γ daria a relação entre a profundidade nos referenciais de I_r e I_{rR} , como mostra a Equação 3.37. Os pixels inicial (u_{iniR}) e final (u_{endR}) no processo de busca pelo pixel correspondente sobre a linha epipolar podem ser calculados como

$$u_{iniR} = \frac{bf_u}{z_{minR}} + u_{rR} \quad (3.38)$$

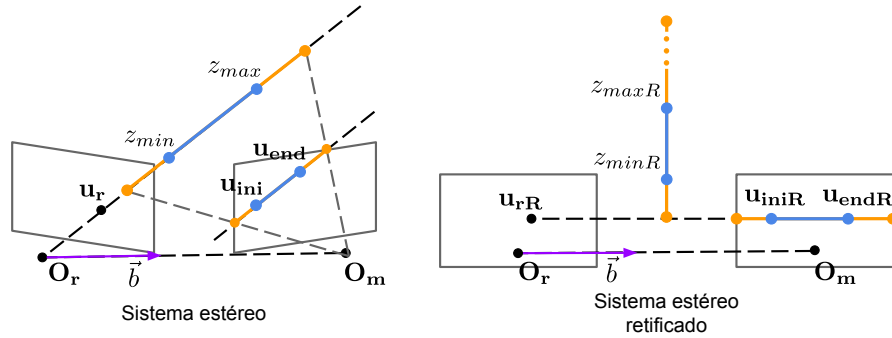
e

⁴ h_{rj} é a j -ésima linha de H_r .

$$u_{endR} = \frac{bf_u}{z_{maxR}} + u_{rR}. \quad (3.39)$$

A Figura 33 apresenta a ideia de redução do universo de busca pela correspondência sobre a linha epipolar, que é delimitada pelos valores de profundidade z_{min} e z_{max} para o sistema estéreo original e z_{minR} e z_{maxR} para o sistema estéreo retificado.

Figura 33 – Delimitação da linha epipolar por z_{min} e z_{max} para o sistema estéreo original e z_{minR} e z_{maxR} para o sistema estéreo retificado.



Fonte: Produção do próprio autor.

Como esse processo de correspondência é feito com 10 imagens, obtêm-se 10 hipóteses de profundidade $z_{r[1-10]}$ para cada pixel \hat{u}_r da imagem de referência.

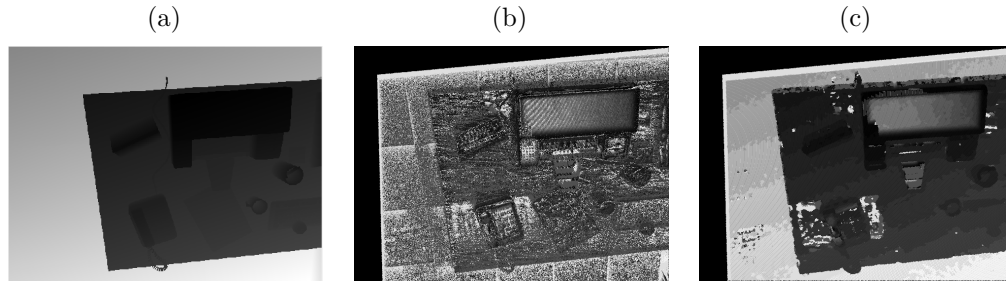
3.3.3 Oclusões, suspixels e precisão sub-pixel

Como mostrado em um exemplo na Seção 3.3.1, as métricas baseadas em janelas apresentam um resultado melhor que quando se utiliza apenas um pixel para encontrar o melhor correspondente. A Figura 34 revisita o resultado obtido no exemplo da Seção 3.3.1.

O resultado de fato é melhor quando se utiliza janelas, entretanto, esse resultado ainda não é igual ao *ground truth*. Um dos motivos disso acontecer é que as métricas baseadas em janela não levam em consideração a profundidade da cena. Na abordagem de janelas se assume que todos os pixels de uma janela serão encontrados na outra imagem em uma janela exatamente igual.

Isso significa dizer que ambas janelas encontram-se no mesmo plano, paralelo ao plano das imagens retificadas. Para que todas as janelas possíveis em uma imagem possam ser encontradas em janelas exatamente iguais em outra imagem, é preciso que a uma imagem seja uma translação da outra no eixo X , ou seja, que a cena seja planar e paralela ao planos das imagens retificadas.

Figura 34 – (a) *Ground truth* e mapas de profundidade da estimação utilizando SAD pixel a pixel (b) e janelas de 7×7 pixels (c). O erro médio absoluto em relação ao *ground truth* de (b) foi de 81,00 *cm* e o de (c) foi de 28,55 *cm*. A área preta que compreende o canto superior esquerdo representa os pixels que não foram reconstruídos por serem pontos da cena que não estavam na imagem 2.



Fonte: Produção do próprio autor.

Entretanto, uma cena é formada por diversos tipos de superfície, sendo inclusive possível encontrar discontinuidades, como o caso da transição entre o abajur e os livros do fundo na Figura 35b.

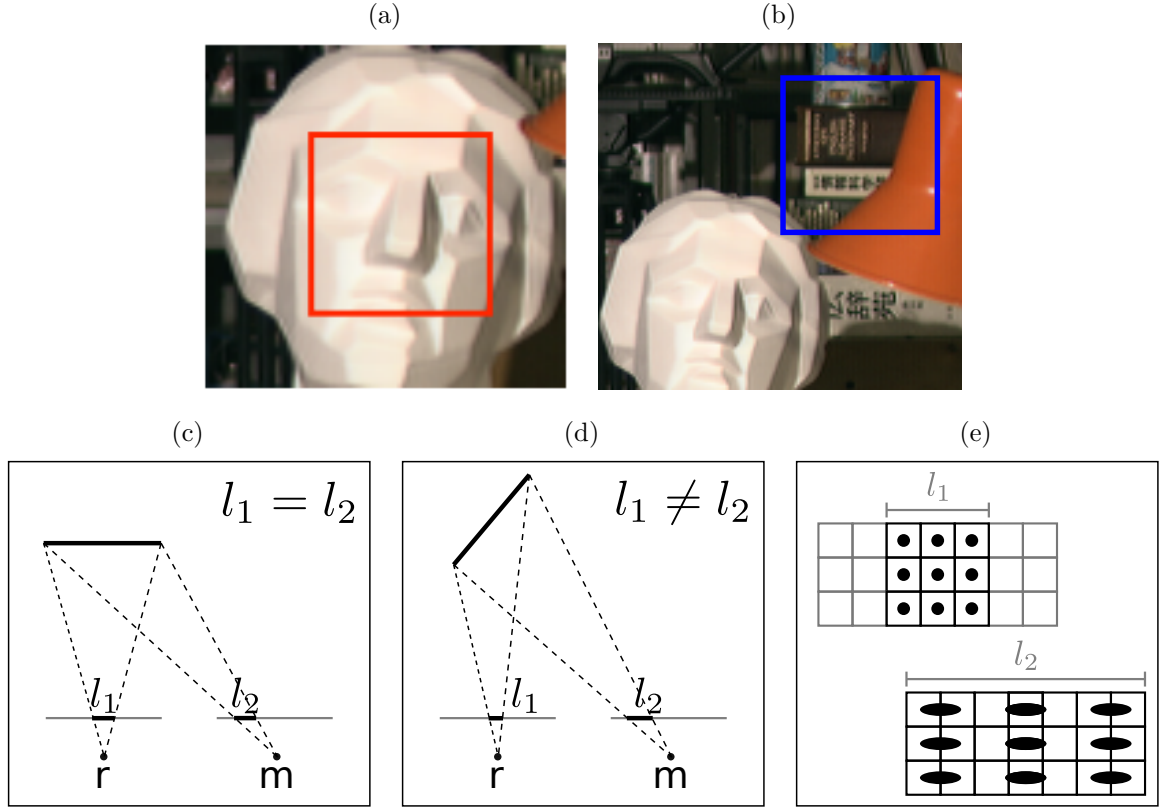
Para contemplar esses casos, a janela precisa levar em conta a curvatura da superfície. No plano da imagem, isso aparece como uma distorção da janela, de forma que os pontos comparados em ambas imagens sejam os mesmos. A Figura 35c apresenta o caso em que a superfície compreendida pela janela das imagens é paralela ao plano das imagens retificadas. Nesse caso é possível notar que as janelas possuem o mesmo tamanho nas duas imagens. Já o caso de uma superfície plana inclinada em relação à *baseline* não apresenta o mesmo resultado (Figura 35d). A janela de uma das imagens da Figura 35d sofreu uma deformação assim como apresentado na Figura 35e para que os pontos comparados correspondessem corretamente.

Pode parecer que deformar as janelas é a solução para esses problemas, a resposta para levar ao resultado ótimo. No entanto, para saber qual a deformação de cada janela é necessário conhecimento prévio da profundidade dos pixels, que é exatamente o que se deseja descobrir (“problema da galinha e do ovo”) (NAVAB, 2010).

O outro problema citado são os casos onde a janela apresenta pixels com descontinuidade da profundidade. Isso é abordado na Figura 36.

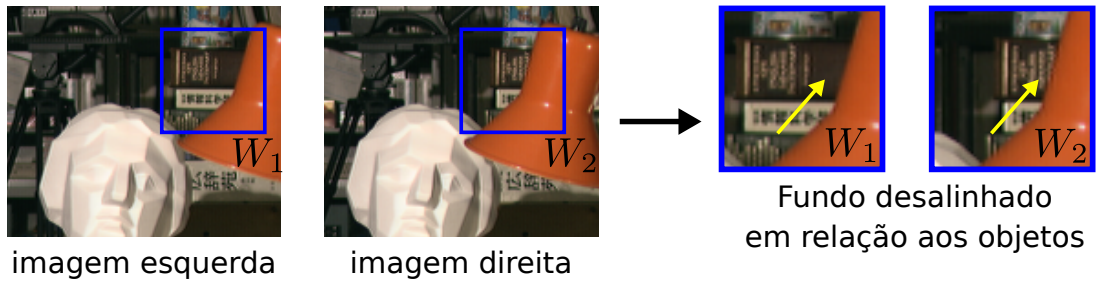
Pode-se notar nessa figura que os pixels das janelas não coincidem. Isso acontece justamente pela descontinuidade da profundidade da superfície contida nas janelas. As Figuras 35a, 35c, 35d e 35e apresentam casos que compreendem superfícies contínuas, mesmo que tendo curvas ou certa inclinação em relação ao plano das imagens retificadas. Independentemente da região do fundo ser plana, no plano da imagem, pontos mais próximos da câmera deslocam mais que os localizados longe dela. Matematicamente, essa

Figura 35 – (a) a janela vermelha apresenta uma superfície não plana. (b) a janela azul apresenta uma descontinuidade da profundidade da cena na transição entre o abajur e os livros do fundo. (c) e (d) são casos onde a superfície compreendida pela janela nas imagens é paralela e inclinada ao plano retificado, respectivamente. (e) representação da janela ideal de (d), onde os pixels comparados (pontos pretos) coincidem.



Fonte: Produção do próprio autor.

Figura 36 – Exemplo de oclusão no par de imagens estéreo Tsukuba. A área apontada pela seta amarela em W_1 sofre oclusão pelo abajur laranja em W_2 .



Fonte: Navab (2010) (adaptado pelo autor).

relação foi apresentada na Seção 3.3.2 pela Equação 3.40

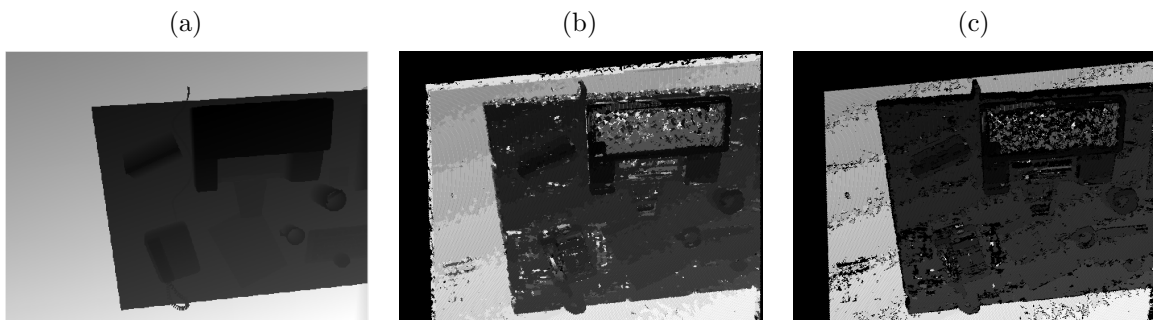
$$z_R = \frac{bf_u}{u_{rR} - u_{mR}}. \quad (3.40)$$

Desse modo, pode-se notar que a profundidade z_R é inversamente proporcional à disparidade. Quanto mais próximo o ponto está da câmera, menor a sua profundidade e, conseqüentemente, maior o deslocamento ($u_{rR} - u_{mR}$) do pixel nas imagens.

Devido a isso, acabam ocorrendo oclusões na imagem por regiões mais próximas da câmera, como é o caso do abajur laranja (Figura 36). Ele ocluiu a região homogênea do livro marrom que se encontra na estante, não sendo possível encontrá-la na outra imagem. Isso pode levar a valores errados nas métricas, não sendo possível identificar corretamente o correspondente na outra imagem. O que geralmente acontece é que os pixels de borda da região ocluída são reconstruídos com mesma profundidade dos pixels de borda da outra região por causa da janela. Em outras palavras, pixels que sofrem oclusão não devem ser comparados, já que não se encontram nas duas imagens.

No trabalho aqui proposto foi utilizada uma técnica chamada de consistência esquerda-direita (*Left-Right Check*), também conhecida na literatura como consistência mútua (AIT-JELLAL; ZELL, 2015). Basicamente, a busca pela correspondência é feita considerando-se primeiramente I_r como imagem de referência e depois I_m . Dessa forma, se a correspondência dos dois casos for a mesma, essa estimacão é considerada como válida, do contrário, o resultado é descartado. A Figura 37 apresenta um exemplo comparativo da estimacão de profundidade no caso em que se confere a consistência mútua e no caso em que isso não é feito.

Figura 37 – (a) *Ground truth* e mapas de profundidade da estimacão utilizando NCC (janelas de 7×7 pixels) sem (b) e com (c) verificacão da consistência esquerda-direita. O erro médio absoluto em relacão ao *ground truth* de (b) foi de 22,37 cm e o de (c) foi de 6,56 cm. As áreas pretas representam pixels que não foram reconstruídos ou por serem pontos da cena que não estavam na imagem 2 ou por serem dados como inválidos. O percentual total de pixels reconstruídos em (b) foi de 83 % e em (c) de 66 %.



Fonte: Produção do próprio autor.

É possível observar na Figura 37 que quando se confere a consistência mútua, o erro médio absoluto cai quase quatro vezes. Apesar de reconstruir 17 % a menos dos pixels da imagem, essa é uma informacão que poderia levar a um resultado incorreto de estimacão

final. A desvantagem desse método é o custo computacional que praticamente dobra, por ser necessário realizar a busca na linha epipolar em ambas imagens de cada par.

Como pode-se notar na Figura 37c, muitas das correspondências incorretas provenientes da ambiguidade de regiões homogêneas também foram descartadas. Entretanto, ainda sobram alguns que podem levar a uma estimacão equivocada de profundidade. Muitos correspondentes incorretos propagam para a etapa de densificação da nuvem de pontos (etapa de tratamento de regiões homogêneas, Seção 3.4) como *outliers*, o que para a reconstrução final é bem ruim.

Para retirar essas más correspondências, pode-se conferir a variância de intensidade dos pixels das janelas de I_r . Caso a variância de cada canal apresente valores abaixo de um limiar, a janela é considerada homogênea, e o pixel não será reconstruído (esses pixels cuja estimacão de profundidade é duvidosa/suspeita, serão chamados de **suspixels**⁵). Isso economiza tempo de processamento, visto que percorrer a linha epipolar e calcular a métrica para cada possível correspondente é uma operação muito custosa no processo de reconstrução 3D.

A variância de cada canal das janelas de I_r seria calculada como

$$Var(W) = \frac{1}{n} \sum_{j=0}^{n^2-1} (w_j - \bar{W})^2, \quad (3.41)$$

onde, para facilitar a notação da equação, W representa um canal da janela W_r de $n \times n$ pixels, \bar{W} a média da intensidade dos pixels de W , e w_j é o j – ésimo pixel desse canal.

Entretanto, para não aumentar muito o custo computacional, no trabalho aqui proposto, o conceito de variância foi calculado como

$$Var(W) = \frac{1}{n} \sum_{j=0}^{n^2-1} |w_j - \bar{W}|. \quad (3.42)$$

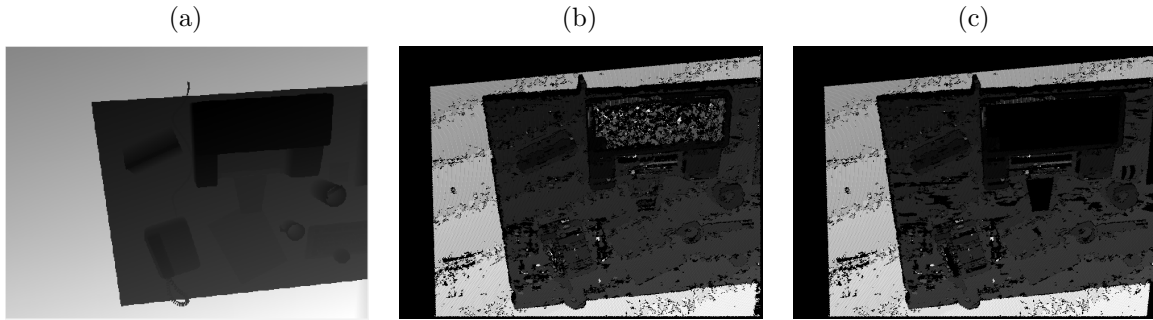
Dessa forma, a Figura 38 apresenta um exemplo comparativo da estimacão de profundidade no caso em que se confere se as janelas de I_r são homogêneas e no caso em que isso não é feito.

Pode-se notar que há uma diminuição do erro médio absoluto com a retirada dos pontos não confiáveis (suspixels). Mas o grande ganho dessa etapa não é fácil de se notar nos mapas de profundidade apresentados, mas pode-se ver na nuvem de pontos gerada no final da etapa de reconstrução esparsa (z_s), como apresentado na Figura 39.

Geralmente, os suspixels não possuem uma boa estimacão de profundidade, como pode-se notar na tampa da impressora da Figura 39a. A superfície da tampa da impressora

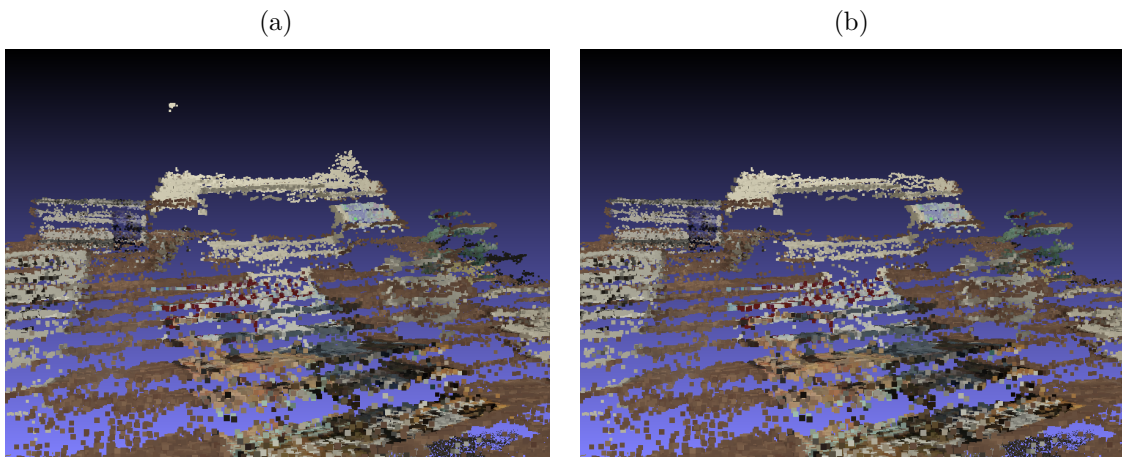
⁵ O termo suspixel foi criado durante o desenvolvimento deste trabalho.

Figura 38 – (a) *Ground truth* e mapas de profundidade da estimação utilizando NCC com verificação esquerda-direita (janelas de 7×7 pixels), sem (b) e com (c) verificação de suspixels. O erro médio absoluto em relação ao *ground truth* de (b) foi de $6,56\text{ cm}$ e o de (c) foi de $5,66\text{ cm}$. O percentual total de pixels reconstruídos em (b) foi de 66% e em (c) de 60% .



Fonte: Produção do próprio autor.

Figura 39 – Nuvem de pontos referente a z_S para o caso sem (a) e com (a) verificação de suspixels.

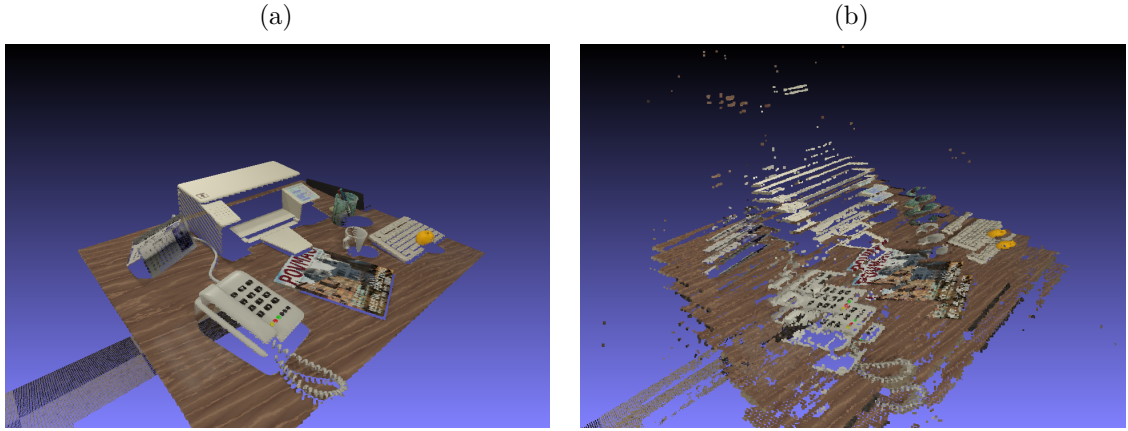


Fonte: Produção do próprio autor.

deveria ser plana, e como pode-se notar nessa figura, ela apresenta uma elevação. Essa elevação pode dificultar a obtenção de uma boa estimativa final de profundidade. A Figura 39b mostra que muitos desses pontos foram descartados. Já os demais com estimativa incorreta serão tratados na etapa de filtragem, explicitada na Seção 3.3.5.

Até então todo o processo descrito pelas Seções 3.3.1, 3.3.2 e 3.3.3 tratou de cálculos discretos de disparidade, ou seja, de profundidade. Entretanto, o mundo é contínuo, a profundidade varia suavemente, salvo descontinuidades. E por isso, apesar da Figura 38c apresentar um resultado parecido com o *ground truth*, essas imagens só apresentam 256 valores diferentes de intensidade de pixel, não sendo possível ver todas as irregularidades da estimativa. A Figura 40 possibilita ver esse problema.

Figura 40 – Nuvem de pontos do *ground truth* (a) e do resultado apresentado na Figura 38c (b).

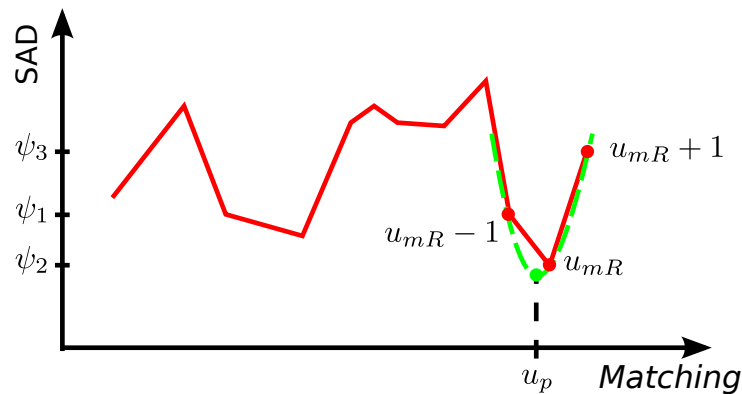


Fonte: Produção do próprio autor.

A Figura 40b apresenta a nuvem de pontos do resultado apresentado na Figura 38c e a Figura 40a do *ground truth*. Como pode ser observado, a estimacão encontrada apresenta vários níveis de profundidade, sendo cada um paralelo ao plano da imagem, refletindo o passo discreto da busca sobre as linhas epipolares.

Segundo Navab (2010), para se obter resultados com uma precisão sub-píxel, pode-se aproximar a função de métrica localmente utilizando uma parábola, como mostra a Figura 41. Os três pontos que formam a parábola são o píxel retornado pela técnica *winner-takes-all* e seus dois vizinhos.

Figura 41 – Aproximação local da função de métrica por uma parábola. Os três pontos que formam a parábola são o píxel retornado pela técnica *winner-takes-all* e seus dois vizinhos.



Fonte: Navab (2010) (adaptado pelo autor).

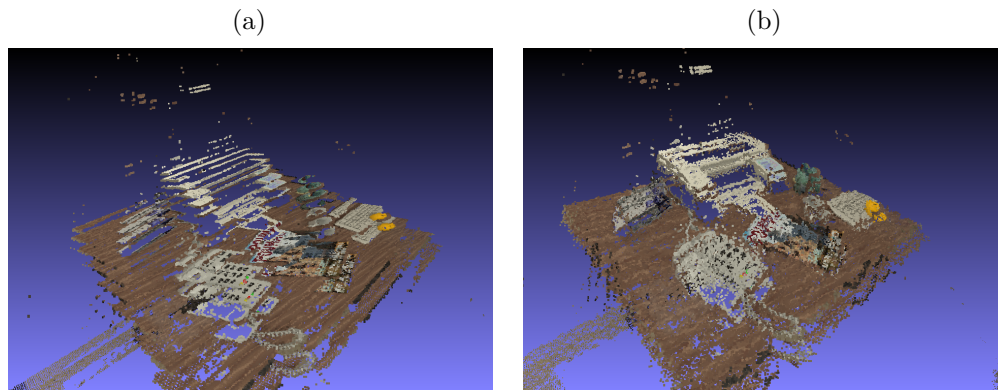
Dessa forma, dado que os píxeis que formam a parábola são u_{mR} , $u_{mR}-1$ e $u_{mR}+1$ e seus respectivos valores na função de métrica iguais a ψ_2 , ψ_1 e ψ_3 , o valor do ponto de

mínima da parábola u_p pode ser calculado como

$$u_p = u_{mR} + \frac{\psi_3 - \psi_1}{2(2\psi_2 - \psi_1 - \psi_3)}. \quad (3.43)$$

A Figura 42 apresenta o resultado mostrado na Figura 38c antes e depois de se realizar essa aproximação.

Figura 42 – Nuvem de pontos do resultado apresentado na Figura 38c antes (a) e depois (b) de se realizar a aproximação local da função de métrica por uma parábola. O erro médio absoluto em relação ao *ground truth* de (a) foi de 5,66 cm e o de (b) foi de 5,46 cm.



Fonte: Produção do próprio autor.

Como pode-se notar na Figura 42, os pixels ficaram mais distribuídos, tomando forma de uma superfície mais contínua. A aparência ruidosa pode levar à conclusão de que o resultado está pior, entretanto, os níveis planos da Figura 42a enganam o leitor, visto que os pixels reconstruídos incorretamente sobre um plano podem passar a impressão de um resultado mais preciso já que planos são superfícies suaves. Essa inversão de percepção pode ser confirmada na diminuição do erro médio absoluto quando se usa a aproximação local da função de métrica por uma parábola.

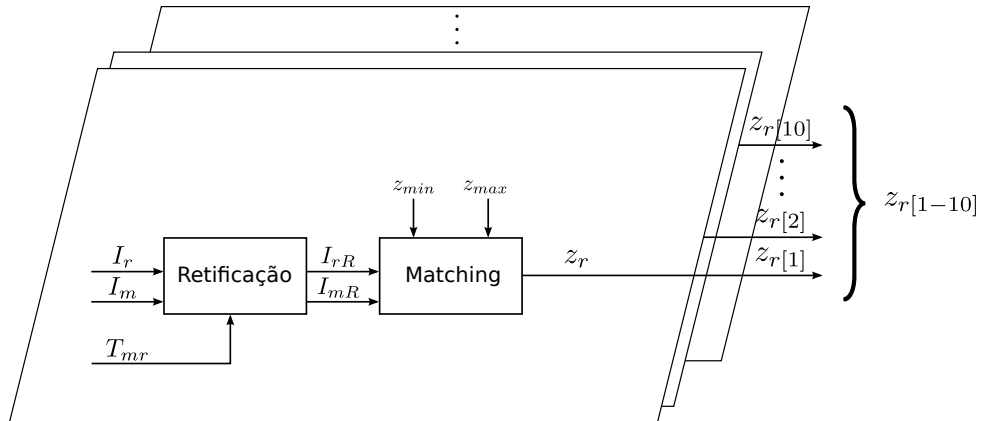
Resultados mais precisos podem ser encontrados na etapa de filtragem que será explicada na Seção 3.3.5. Todos os tratamentos discutidos na seção atual fazem parte apenas da etapa de *matching*.

3.3.4 Pirâmide gaussiana

As Seções 3.3.1, 3.3.2 e 3.3.3 descrevem o processo apresentado na Figura 43.

Entretanto, esse processo ainda é bastante custoso. Por isso, para tentar acelerar o processo, foi utilizada uma abordagem de pirâmides de multiresolução, também conhecida como *coarse-to-fine* (aproximação grosseira para fina) (TANSKANEN et al., 2013). Essa

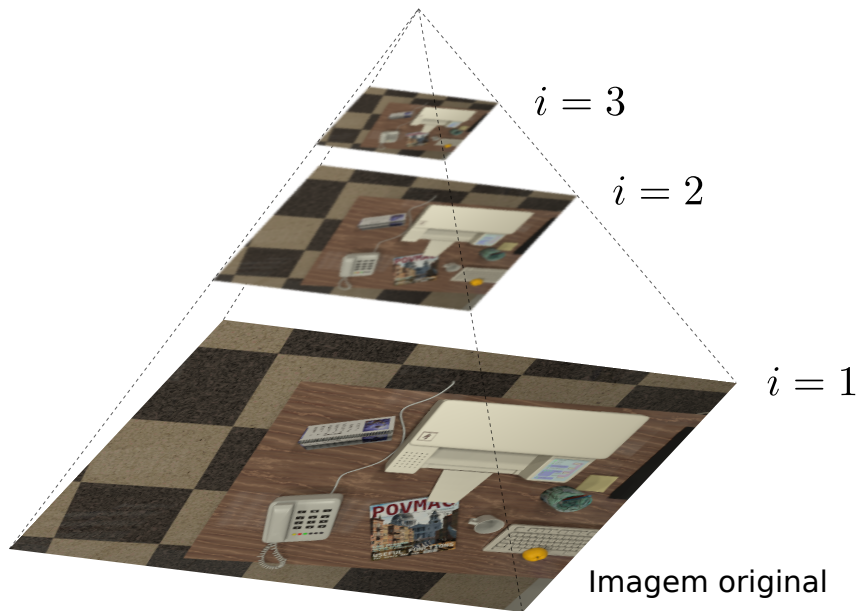
Figura 43 – Esquemático do processo descrito pelas Seções 3.3.1, 3.3.2 e 3.3.3.



Fonte: Produção do próprio autor.

abordagem subamostra as imagens utilizando pirâmides gaussianas⁶, como apresentado na Figura 44.

Figura 44 – Abordagem *coarse-to-fine* utilizando pirâmides gaussianas. O número P de níveis i nesta figura é igual a 3.



Fonte: Produção do próprio autor.

A cada nível $i \in \{1, \dots, P\}$ da pirâmide, sua respectiva imagem é obtida diminuindo-se a resolução de sua versão no nível $i-1$ pela metade. Entretanto, antes desse procedimento, a versão do nível $i-1$ deve sofrer uma filtragem passa-baixas (filtro gaussiano) para evitar o *aliasing*. A imagem original, de maior resolução, se encontra na base, onde $i = 1$. A

⁶ Para encontrar cada nível da pirâmide foi utilizada a função `pyrdown()` da biblioteca OpenCV (BRADSKI, 2000).

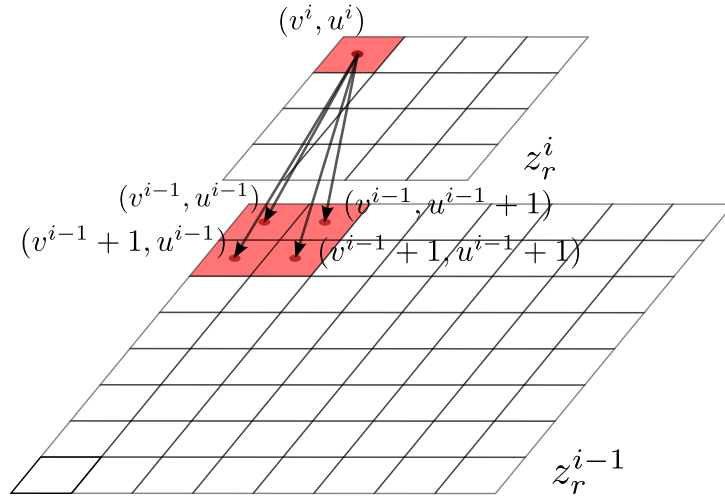
construção das subamostras em uma pirâmide é feita para cada par (I_r, I_m) antes de se iniciar a reconstrução esparsa.

Começando pelo nível mais alto da pirâmide, realiza-se o processo apresentado na Figura 43, retornando a matriz z_r^i (de mesma dimensão que I_r do nível i), contendo a profundidade estimada para cada pixel reconstruído. A seguir, essa matriz é sobreamostrada de forma que a profundidade do pixel (u^i, v^i) do nível i propague-se para seu correspondente no nível $i - 1$ e três de seus vizinhos

$$\begin{aligned} z_r^{i-1}(v^{i-1}, u^{i-1}) &= z_r^i(v^i, u^i) \\ z_r^{i-1}(v^{i-1} + 1, u^{i-1}) &= z_r^i(v^i, u^i) \\ z_r^{i-1}(v^{i-1}, u^{i-1} + 1) &= z_r^i(v^i, u^i) \\ z_r^{i-1}(v^{i-1} + 1, u^{i-1} + 1) &= z_r^i(v^i, u^i), \end{aligned} \quad (3.44)$$

onde $(u^i, v^i) := (\lfloor u^{i-1}/2 \rfloor, \lfloor v^{i-1}/2 \rfloor)$, sendo $\lfloor x \rfloor$ o maior inteiro menor ou igual a x , assim como mostra a Figura 45.

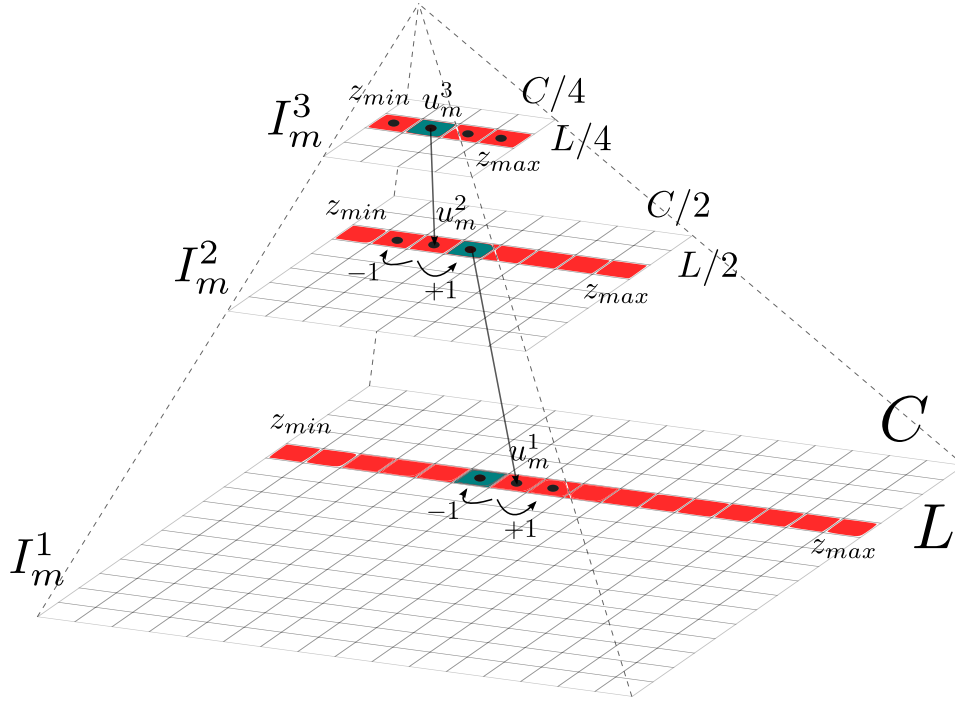
Figura 45 – Sobreamostragem de z_r para o processo de *matching* do próximo nível da pirâmide.



Fonte: Produção do próprio autor.

A partir de z_r^{i-1} pode-se calcular qual o correspondente u_m^{i-1} relativo a essa profundidade. Neste trabalho, considera-se que o resultado advindo do nível anterior possui uma incerteza de um pixel no nível atual. Dessa forma, no primeiro nível se busca por toda a linha epipolar, definida de z_{min} a z_{max} , mas nos níveis posteriores, a busca é feita somente na vizinhança do correspondente indicado pelo nível anterior, como apresentado na Figura 46.

Figura 46 – Busca pela linha epipolar. No primeiro nível se busca por toda a linha epipolar, definida de z_{min} a z_{max} . Nos níveis posteriores, a busca é feita somente na vizinhança do correspondente indicado pelo nível anterior. As setas apontam o pixel no nível corrente relativo ao resultado do nível anterior (u_m^i). Os pixels em vermelho apresentam a linha epipolar de z_{min} a z_{max} e os de azul o resultado do *matching* no nível corrente. Os pontos pretos apresentam a linha epipolar definida pelos limites de cada nível.



Fonte: Produção do próprio autor.

u_m^{i-1} refere-se aos pixels em I_m^{i-1} correspondentes a u_r^{i-1} cuja profundidade é igual a z_r^{i-1} .⁷ Os limites dados por $u_m^{i-1} - 1$ e $u_m^{i-1} + 1$ ainda respeitam a faixa (z_{min}, z_{max}) . Dessa maneira, a complexidade da etapa de *matching*, ou seja, o número de operações de métrica por par de imagem para o pior caso, que antes era

$$complexidade = LCN, \quad (3.45)$$

onde L é o número de linhas da imagem original e C o número de colunas (LC é o número total de pixels da imagem original), N é o número de pixels da linha epipolar no processo sem pirâmide, passa a ser

$$complexidade = \frac{LCN}{2^{3(P-1)}} + \sum_{i=0}^{P-2} \frac{3LC}{2^{2i}}, \quad (3.46)$$

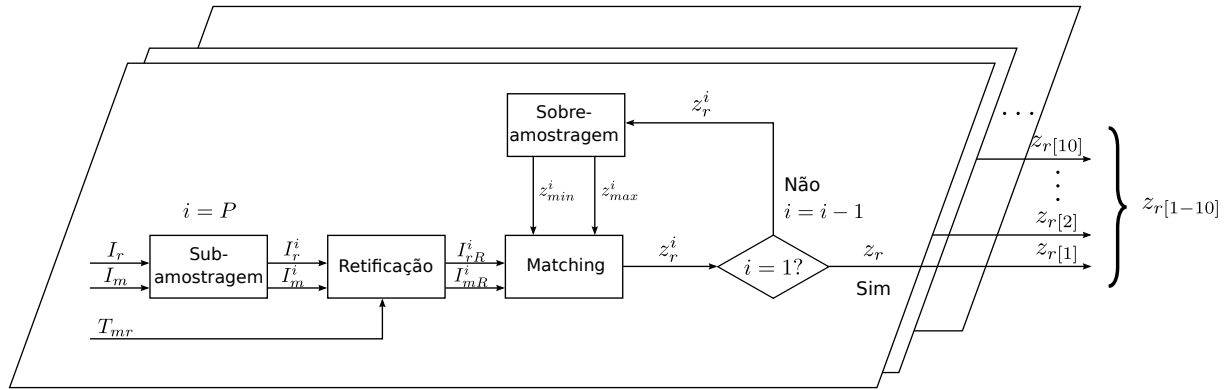
com P sendo o número de níveis da pirâmide.

⁷ I_1^i refere-se à versão da imagem I_1 no nível i da pirâmide.

Como pode-se notar nas Equações 3.45 e 3.46, quanto maior o número de níveis da pirâmide, menor a complexidade do processo, maior a velocidade de reconstrução. A desvantagem dessa abordagem é que quanto mais níveis se usa, menor fica a imagem do nível P , e menos informação ela terá, visto que detalhes pequenos, mesmo que marcantes, são perdidos a cada nível que se cria. Além disso, pixels não reconstruídos no nível i são propagados para quatro pixels que não serão reconstruídos no nível $i - 1$, levando a um possível número maior de pixels não reconstruídos.

O final desse processo retorna uma matriz z_r de dimensão $L \times C$ contendo a profundidade de cada pixel reconstruído. A Figura 43 que apresenta o algoritmo sem o processo de pirâmide pode ser alterada de forma a incorporar essa abordagem, como mostra a Figura 47.

Figura 47 – Esquemático do processo descrito pelas Seções 3.3.1, 3.3.2, 3.3.3 e 3.3.4 com a abordagem de pirâmide incorporada.



Fonte: Produção do próprio autor.

Como esse processo de correspondência é feito com 10 pares de imagens, obtêm-se 10 hipóteses de profundidade $z_{r[1-10]}$ para cada pixel $\hat{\mathbf{u}}_r$, como já mencionado na Seção 3.3.2.

3.3.5 Filtragem dos resultados

Mesmo com as soluções propostas na Seção 3.3.3, o sistema proposto não descarta todos os pixels que apresentam uma estimacão incorreta de profundidade. Além disso, devido ao CCD possuir um número de sensores finito, a solução fica a mercê de saltos discretos de possibilidades de correspondência, sendo necessário algum tipo de aproximação, como o da parábola, para se obter valores em sub-pixels.

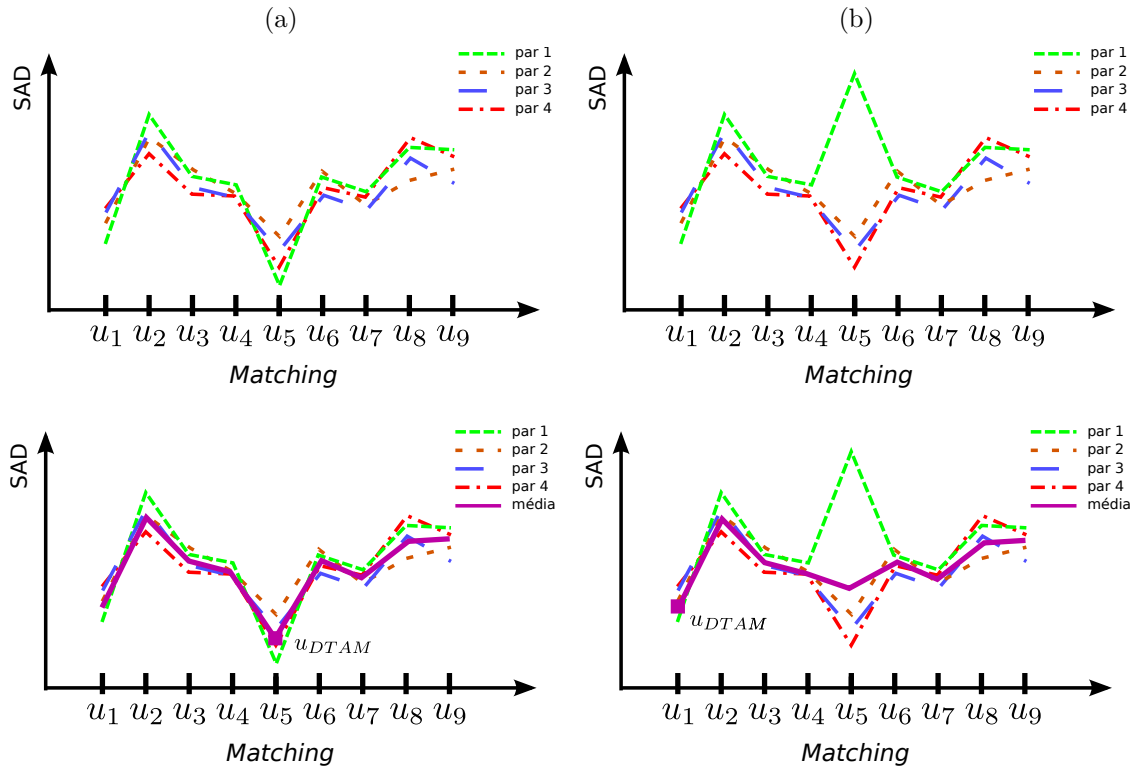
A solução proposta pelo DTAM foi de se usar mais de um par de imagens para formar a função de métrica⁸, tentando, assim, filtrar ruídos encontrados em certos pares (I_r, I_m) que não aparecem em outros. Esse conjunto de reconstruções, ao invés de se utilizar

⁸ Como mostrado em (NEWCOMBE; LOVEGROVE; DAVISON, 2011), para se obter um resultado adequado, o DTAM utiliza pelo menos 30 imagens com certa sobreposição.

apenas um par de imagens, serve como redundância de informações para que se possa retirar os valores discrepantes dos dados.

Quando se tirar a média das funções, o efeito de *outliers* será amenizado, entretanto, se um ou mais *outliers* (ruído) forem muito fortes, isso afetará essa média, o que poderá, mesmo com esse filtro, levar a valores incorretos de estimativa, como mostra a Figura 48.

Figura 48 – funções de métrica SAD provenientes de 4 diferentes pares de imagem (I_r, I_m) sobre a linha epipolar (de u_1 a u_9) de um mesmo pixel \mathbf{u}_r . Cada par de imagens possui certo ruído na intensidade dos seus pixels, um dos motivos pelo qual as curvas não coincidem. Funções com ruído (a) fraco e (b) forte (gráficos superiores) e a média dessas funções em roxo (gráficos inferiores), resultando no correspondente u_{DTAM} , sendo u_5 a correspondência correta.



Fonte: Produção do próprio autor.

Isso explica a quantidade de imagens utilizadas pelo DTAM. Quanto maior o número de pares de imagens reconstruídos, menor será o efeito do ruído. Como mostrado em (NEWCOMBE; LOVEGROVE; DAVISON, 2011), o melhor resultado do DTAM foi obtido a partir de um conjunto de 30 imagens, sendo que foram testados conjuntos de 2, 10 e 30 imagens.

A abordagem escolhida neste trabalho para fundir os resultados foi baseada na estratégia proposta em (CONCHA; CIVERA, 2015) (consistência temporal) que, diferentemente do DTAM, usa apenas 10 pares de imagens, e consegue obter resultados melhores. Além disso, como apresentado na Seção 3.3.3, *outliers* são removidos para resta-

rem apenas pontos dados como confiáveis, enquanto no DTAM, todos os pixels tentarão ser reconstruídos, mesmo que o resultado final seja errado. Os resultados dos 10 pares $z_{r[1-10]}$ são fundidos e filtrados por meio de dois procedimentos: a consistência temporal e a consistência espacial.

Como comentado no início deste capítulo, o inverso de profundidade ξ é utilizado nessa etapa de fusão e filtragem ao invés da profundidade z pela discretização ser feita de forma linearmente espaçada em termos de pixels (disparidade) e não da profundidade, para que a linha epipolar seja varrida uniformemente. Dessa forma, $\frac{1}{z} = \xi \propto \text{disparidade}$, logo, $\xi_{u[1-10]} = \frac{1}{z_{r[1-10]}}$.

Na etapa de **consistência temporal**, uma estimativa de profundidade será considerada aceitável (*inlier*) caso as hipóteses do inverso de profundidade ξ de várias imagens sejam semelhantes. Portanto, as 10 hipóteses de inverso de profundidade de cada pixel $\xi_{u[1-10]}$ são comparadas buscando-se por valores compatíveis entre pelo menos 5 das 10 hipóteses, testando-se todas as combinações possíveis.

Esse conjunto de valores compatíveis devem respeitar a relação

$$\frac{(\xi_{u[j,j+n]}^{\max} - \xi_{u[j,j+n]}^{\min})}{\sigma_{\xi_u}} < 2, \quad (3.47)$$

onde a diferença entre o máximo $\xi_{u[j,j+n]}^{\max}$ e mínimo $\xi_{u[j,j+n]}^{\min}$ inverso de profundidade dessa combinação deve ser menor que dois desvios padrão σ_{ξ_u} da combinação.

σ_{ξ_u} é o desvio padrão da combinação, sendo calculado em função do desvio padrão de cada hipótese da combinação

$$\sigma_{\xi_u} = \sqrt{\left(\sum_{k=j}^{j+n} \frac{1}{\sigma_{\xi_u[k]}^2} \right)^{-1}}, \quad (3.48)$$

onde $\sigma_{\xi_u[k]}$ é o desvio padrão da hipótese k . O desvio padrão de cada hipótese $\sigma_{\xi_u[1-10]}$ é obtido de forma aproximada. Assume-se que $\xi_{u[1-10]}$ possui uma incerteza de um pixel, assim,

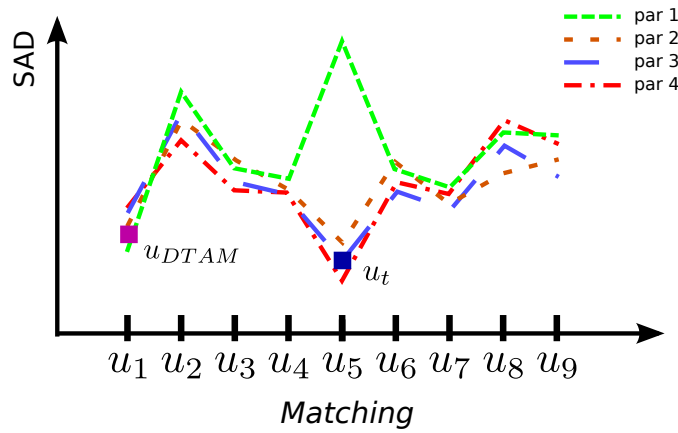
$$\sigma_{\xi_u[1-10]} = \xi_{u[1-10]} - \xi_{(u+1)[1-10]}, \quad (3.49)$$

para $\xi_{(u+1)[1-10]}$ sendo as 10 hipóteses de inverso de profundidade do pixel vizinho a u na linha epipolar.

Esse teste é repetido para $n = 4, \dots, 9$ e $j = 1, \dots, 10 - n$, abrangendo todas as diferentes combinações de hipóteses possíveis. O inverso de profundidade ótimo ξ_u resultante da consistência temporal é a média da combinação de hipóteses do maior conjunto que satisfaz as Equações 3.47 3.48 e 3.49. A Figura 49 apresenta um exemplo

de resultado da consistência temporal para os valores da função de métrica apresentados na Figura 48b, desconsiderando o fato de que são necessárias no mínimo 5 hipóteses semelhantes para se ter um conjunto válido (o exemplo da Figura 48b só apresenta 4 funções de métrica, ou seja, 4 hipóteses).

Figura 49 – Exemplo de resultado da consistência temporal u_t para os valores da função de métrica apresentados na Figura 48b. Nesse exemplo se desconsidera o fato de que são necessárias pelo menos 5 hipóteses semelhantes para se ter um conjunto válido. O quadrado azul apresenta o resultado da consistência temporal u_t , e o roxo apresenta o resultado u_{DTAM} do DTAM dado pelo exemplo da Figura 48b.



Fonte: Produção do próprio autor.

Note que o resultado do *matching* obtido com as métricas 2, 3 e 4 indicam o quinto pixel como o de maior similaridade, enquanto que a métrica 1 indica o primeiro pixel. Sem o filtro temporal, o DTAM escolheria u_1 como melhor *matching*. Entretanto, usando-se a consistência temporal, u_5 será corretamente escolhido.

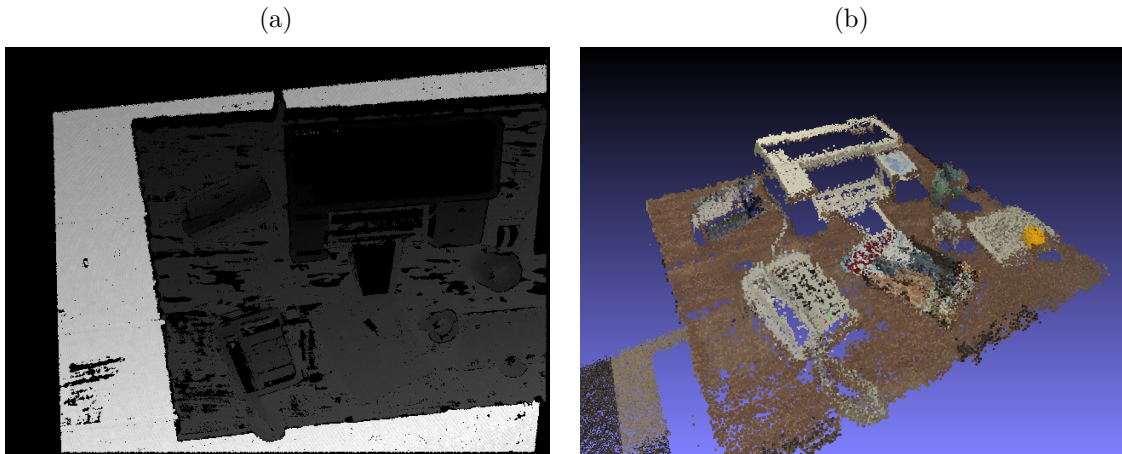
No final dessa etapa, a profundidade resultante da consistência temporal z_t é obtida ao se inverter ξ_u .

A Figura 50 apresenta o mapa de profundidade pós etapa de consistência temporal para um conjunto de 10 imagens e uma de referência, sendo o resultado de uma das hipóteses apresentado na Figura 42b. Nesse exemplo, a métrica utilizada foi o NCC, com janela de 7×7 pixels com verificação esquerda-direita e de suspixels, e com aproximação local por parábola.

A Figura 50a apresenta menos pontos claros em regiões escuras que a Figura 38c, ou seja, menos pontos com estimativa incorreta na nuvem de pontos. O erro absoluto médio mais baixo reflete esse resultado. Entretanto, aparentemente pelo mapa de profundidades da Figura 50a não há ruídos no resultado, o que não é verdade. Esse ruído pode ser notado na Figura 50b com uma característica conhecida como **ruído sal e pimenta**.

Assume-se que a profundidade no mundo varia suavemente, portanto, pixels vizinhos

Figura 50 – (a) Mapa de profundidade pós etapa de consistência temporal para métrica NCC com verificação esquerda-direita (janelas de 7×7 pixels) com verificação de suspixels, e com aproximação local por parábola. (b) Nuvem de pontos do resultado de (a). O erro médio absoluto em relação ao *ground truth* foi de 2,35 cm contra 5,46 cm da Figura 42b, com percentual total de pixels reconstruídos de 60 %.



Fonte: Produção do próprio autor.

devem apresentar profundidade semelhante. Para retirar esse ruído sal e pimenta pode-se utilizar um **filtro de mediana**. Desta forma, ao se aplicar um filtro de mediana na matriz de estimativas de profundidade z_t dos pixels que não foram descartados durante os procedimentos de fusão da consistência temporal o resultado é retornado na matriz z_s . Essa etapa de filtragem é chamada de **consistência espacial**.

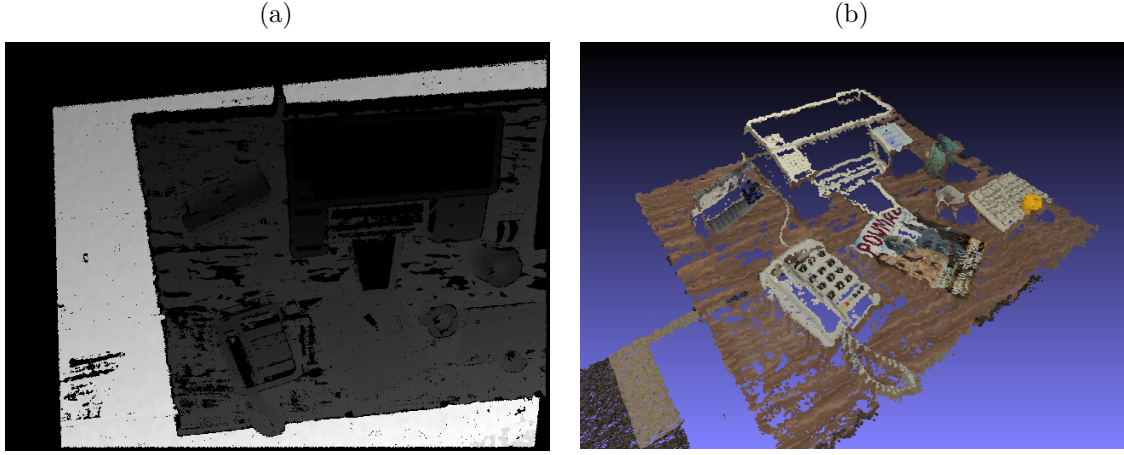
O mapa de profundidade e a nuvem de pontos desse resultado suavizado pelo filtro de mediana pode ser conferido na Figura 51.

Durante todo o processo de reconstrução os pixels dados como *outliers* (descartados) são anotados para que se possa saber na etapa de tratamento de regiões homogêneas quais pixels de z_s possuem valor com significado.

A Figura 52 apresenta o esquemático do algoritmo de reconstrução esparsa descrito pelas Seções 3.3.1 a 3.3.5.

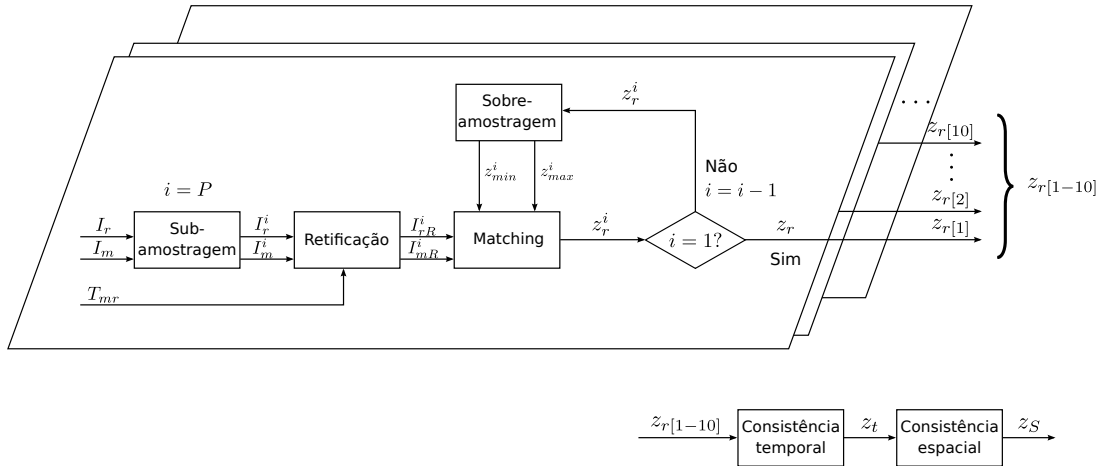
Como pode-se ver na Figura 52, os resultados z_r dos 10 pares de imagens reconstruídos são fundidos e filtrados na etapa de consistência temporal e filtrados novamente na etapa de consistência espacial, resultando na matriz z_s com a estimativa esparsa de profundidade dos pixels de I_r . Tanto a etapa de consistência temporal quanto a de consistência espacial suavizam os resultados de estimativa de profundidade, levando a um resultado mais contínuo, com precisão de sub-pixels.

Figura 51 – Mapa de profundidade (a) e nuvem de pontos (b) do resultado z_S retornado pelo filtro de mediana de 5×5 pixels sobre o resultado apresentado na Figura 50. O erro médio absoluto em relação ao *ground truth* foi de 1,84 cm contra 2,35 cm da Figura 50.



Fonte: Produção do próprio autor.

Figura 52 – Esquemático do algoritmo de reconstrução 3D esparsa descrito pelas Seções 3.3.1 a 3.3.5.



Fonte: Produção do próprio autor.

3.4 Segmentação de regiões homogêneas para densificar a nuvem de pontos

Como pode-se ver nas Figuras 19 e 20 da Seção 3.1, mesmo após o processo completo de reconstrução, a precisão do DTAM acaba sendo muito prejudicada por causa da profundidade estimada das áreas com pouca textura.

Devido à falta de informação em regiões homogêneas, a reconstrução 3D dessas áreas normalmente acumula muito erro. Não é fácil encontrar a correspondência dos pixels

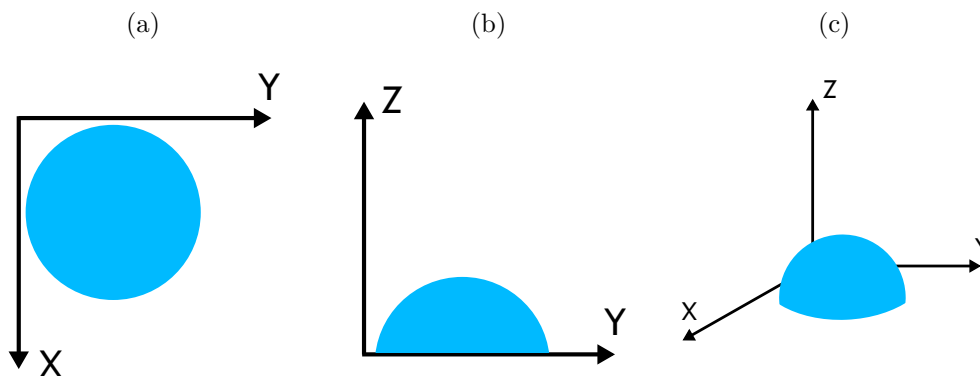
dessas regiões em outras imagens devido à grande semelhança com seus vizinhos.

Pensando nisso, trabalhos como (FRAUNDORFER; SCHINDLER; BISCHOF, 2006; CONCHA; CIVERA, 2015; CONCHA et al., 2015; CONCHA et al., 2014) tentam não apenas estimar a correspondência de pixels entre imagens, mas também extrair informações de entendimento da cena e estimar a correspondência de regiões para aumentar a precisão de seus resultados.

Entretanto, mesmo utilizando informações da cena, ainda fica difícil saber exatamente o formato tridimensional de uma região homogênea. Por isso, estes trabalhos aproximam essas áreas por regiões planares, de forma que, para não estimar erroneamente superfícies curvas, utilizam esse resultado como uma estimativa inicial ou uma hipótese em um processo iterativo de minimização. É importante comentar que o processo iterativo possui limitações, como visto na Seção 3.1. Normalmente se considera suavidade na superfície reconstruída por meio de um termo regularizador na equação a ser minimizada. Todavia, como as regiões foram aproximadas por planos, a superfície já está distribuída de forma suave, sendo difícil de mudar depois para outro formato.

Vale ressaltar que, mesmo assim, esse tipo de consideração não é tão grosseira quanto pode parecer. Pode-se notar, por meio de um exemplo apresentado na Figura 53a que, sem ter qualquer tipo de conhecimento prévio sobre a cena, um ser humano poderia facilmente afirmar que a superfície azul é algum tipo de disco azul ou tapete. Entretanto, ao observar as Figuras 53b e 53c, é possível perceber que o objeto na verdade era uma espécie de calota azul. Ou seja, sem informações a nível de objeto, cena, contexto ou geometria, um robô poderia ter problemas em situações como essa.

Figura 53 – Visão de cima (a), de lado (b) e isométrica de uma calota azul.



Fonte: Produção do próprio autor.

Considerando que as regiões homogêneas correspondem a áreas planas, quaisquer três pontos reconstruídos pertencentes a uma região podem ser utilizados para o cálculo de um plano. Geralmente, os melhores pontos para se calcular os planos fazem parte das

bordas que limitam essas áreas. Isso se deve à falta de textura no interior dessas regiões, dado que pontos com características mais relevantes tem uma chance maior de serem reconstruídos corretamente.

Para aproximar as regiões homogêneas por planos, ou qualquer outro tipo de superfície, primeiramente, é necessário identificá-las na imagem. Esse processo de identificação e separação de regiões da imagem é chamado de segmentação.

Entretanto, é preciso decidir que técnica utilizar para realizar a segmentação das imagens. Li et al. (2015) compara os métodos *Simple Linear Iterative Clustering* (SLIC) de Achanta et al. (2010), SRM (NOCK; NIELSEN, 2004) e o *Superpixels Region Merging* (SPRM) proposto por eles, que junta características dos dois métodos citados. Li et al. (2015) afirma que a técnica SLIC apresenta problemas de sobre-segmentação que podem ser amenizados ao se utilizar métodos de agrupamento estatístico, como o SRM, em Superpixels. Já Fraundorfer, Schindler e Bischof (2006) segmenta regiões homogêneas por meio do algoritmo MSER (*Maximally Stable Extremal Regions*). Os trabalhos de Concha e Civera (2015), Concha et al. (2015) e Concha et al. (2014) utilizam, por outro lado, a técnica de Superpixels proposta por Felzenszwalb e Huttenlocher (2004).

As técnicas MSER (MATAS et al., 2004), Superpixel (FELZENSWALB; HUTTENLOCHER, 2004) e SRM (NOCK; NIELSEN, 2004) foram testadas nas imagens utilizadas neste trabalho afim de encontrar aquela que oferecesse o melhor resultado. As Seções 3.4.1, 3.4.2 e 3.4.3 apresentam o funcionamento de cada uma delas e seus resultados obtidos. As conclusões são apresentadas na Seção 3.4.4. Posteriormente, nas Seções 3.5.1 e 3.5.2 serão explicitadas as duas abordagens desenvolvidas para o tratamento de regiões homogêneas. A primeira considera a aproximação de cada região por planos e a segunda, a aproximação de cada região por uma superfície formada por triângulos planares.

3.4.1 *Maximally Stable Extremal Regions* (MSER)

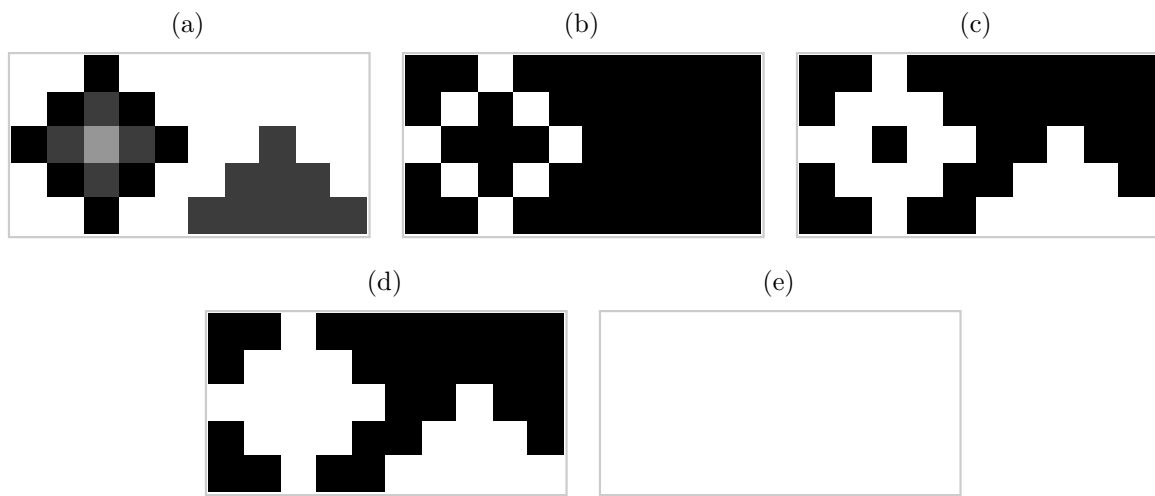
Maximally Stable Extremal Regions, ou MSER, foi um método proposto por Matas et al. (2004) para encontrar a correspondência de regiões entre imagens de um sistema *wide baseline* (sistema cuja a distância entre câmeras não é pequena). Para encontrar corretamente as correspondências, esse método busca **regiões extremas maximamente estáveis** (MSERs).

Nesse método, cada região é um conjunto de pixels conectados. Para que ela seja considerada extrema, todos os pixels da borda externa (pixels que não fazem parte da região mas estão conectados) devem ter intensidade maior que qualquer pixel da região. As regiões extremas são identificadas na imagem por meio de um processo de limiarização, ou seja, todos os pixels (u, v) cuja intensidade $I(v, u)$ seja menor ou igual a um limiar θ são dados como válidos e cada conjunto conectado de pixels válidos na imagem corresponde a

uma região diferente.

Como a intensidade dos pixels pode apresentar 256 valores diferentes em uma imagem com 256 níveis de escala de cinza, é possível aplicar essa mesma quantidade de limiares na imagem. Isso produzirá, conseqüentemente, um conjunto de regiões extremas para cada limiar aplicado, como apresentado na Figura 54. A quantidade de limiares aplicados pode se reduzir controlando-se o passo entre limiares. O parâmetro do algoritmo responsável por esse controle é o Δ .

Figura 54 – Exemplo do processo de limiarização do algoritmo MSER para a busca por regiões extremas ($\Delta = 85$). (a) Imagem em escala de cinza. Resultados da limiarização para (b) $\theta = 0$, (c) $\theta = 85$, (d) $\theta = 170$ e (e) $\theta = 255$.



Fonte: Produção do próprio autor.

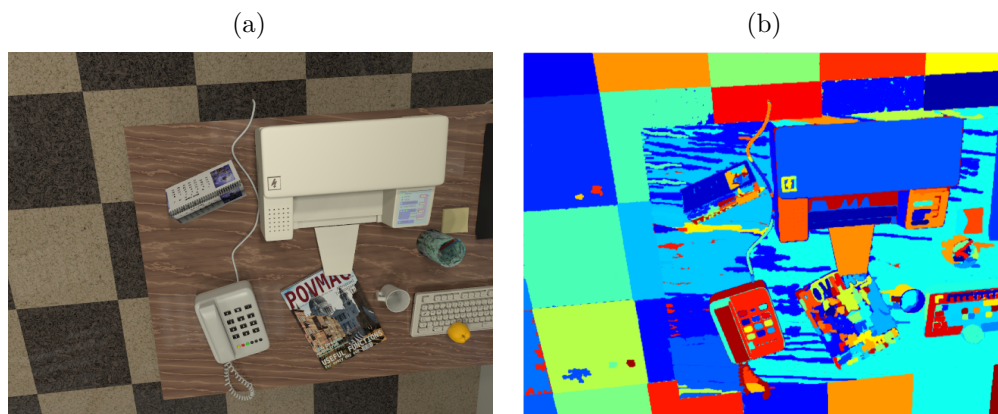
Dentre as regiões extremas encontradas em cada limiar aplicado, somente aquelas cuja área (quantidade de pixels) não apresentou muita alteração de um limiar para o próximo são consideradas **regiões extremas maximamente estáveis** (MSERs). A variação máxima de área permitida para que uma região seja considerada MSER é definida pelo parâmetro '*MaxAreaVariation*' do algoritmo.

Para o exemplo da Figura 54, de $\theta = 0$ para $\theta = 85$ a região extrema encontrada no formato de losango teve um aumento em sua área de 50 %. Caso o parâmetro '*MaxAreaVariation*' fosse de 50 %, essa região (Figura 54c) poderia ser caracterizada como uma MSER. Entretanto, de $\theta = 85$ para $\theta = 170$, essa mesma região apresentou um aumento de área de aproximadamente 8 %, o que também caracteriza a região no formato de losango da Figura 54d como uma MSER. Logo, pode-se notar que essa região gerou duas MSERs que compartilham pixels entre si. A outra região extrema encontrada possui um formato de pirâmide. Essa outra região não varia de área de $\theta = 85$ para $\theta = 170$, portanto, também é qualificada como uma MSER.

Além do passo θ e da variação máxima de área '*MaxAreaVariation*', também pode-se definir um tamanho mínimo e máximo que as regiões devem ter ('*RegionAreaRange*').

A Figura 55 apresenta o resultado da segmentação utilizando o método MSER (MATAS et al., 2004) em uma das imagens utilizadas como imagem de referência no trabalho proposto.

Figura 55 – A imagem (a) foi segmentada utilizando a técnica MSER no Matlab® (b), cujos parâmetros de configuração foram '*ThresholdDelta*' = 2 (Δ), '*RegionAreaRange*' = [30 307200], '*MaxAreaVariation*' = 0.25 e '*ROI*' = [1 1 480 640].



Fonte: Produção do próprio autor.

Como pode-se perceber, diferentemente das outras duas técnicas testadas, o MSER não segmenta de fato as imagens. Essa técnica detecta regiões extremas maximamente estáveis em uma imagem mesmo que duas ou mais regiões compartilhem alguns pixels. A Figura 55b mostra algumas MSERs que compartilham pixels entre si, apesar de não ser fácil de notar pelas regiões terem sido apresentadas todas sobrepostas na mesma imagem. Para facilitar a visualização, o resultado mostrado na Figura 55b pode ser conferido na Figura 56. Dessa vez, somente duas regiões foram apresentadas, onde a região da Figura 56a está contida na região da Figura 56b.

Por isso, essa técnica exige uma etapa a mais, para definir afinal a que região cada pixel pertence. Basicamente, nessa etapa os pixels compartilhados fariam parte somente da região que possui a menor área.

Supondo que a Figura 55b seja o produto dessa etapa adicional, é possível notar um problema comum em qualquer método de segmentação: a sub-segmentação, ou seja, a imagem foi segmentada menos do que deveria ser. Por isso, uma área da mesa e uma do chão foram segmentadas como sendo apenas uma região. Isso pode acarretar em uma aproximação incorreta da profundidade de pixels dessas áreas.

Para uma melhor visualização de cada região, a Figura 57 reapresenta o resultado da segmentação da imagem mostrada na Figura 55b, onde cada cor representa uma região

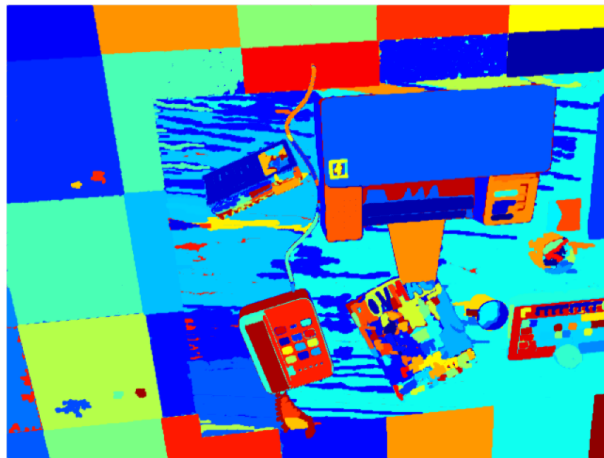
Figura 56 – Resultado da segmentação do método do MSER, onde a região A mostrada em (a) está contida na região B apresentada em (b).



Fonte: Produção do próprio autor.

diferente.

Figura 57 – Resultado da segmentação da imagem mostrada na Figura 55b, onde cada cor representa uma região diferente.



Fonte: Produção do próprio autor.

3.4.2 Statistical Region Merging (SRM)

Statistical Region Merging, proposto por Nock e Nielsen (2004), é um algoritmo rápido e robusto para segmentar imagens em regiões de intensidade de cor similar.

A ideia desse algoritmo é considerar inicialmente cada pixel como uma região e, então, iterativamente, aplicar um teste estatístico em regiões vizinhas para decidir se devem ser fundidas ou não. A Figura 58 apresenta o resultado da segmentação utilizando a técnica SRM em uma das imagens utilizadas como imagem de referência no trabalho proposto.

Figura 58 – A imagem RGB I (a) foi segmentada utilizando a técnica SRM, resultando em I^* (b), cujo parâmetro de configuração $Q = 256$.



Fonte: Produção do próprio autor.

Na Figura 58, a imagem RGB I foi segmentada utilizando a técnica SRM, resultando em I^* . I^* é composta por regiões estatísticas, onde os pixels são representados por distribuições (**pixels estatísticos**), sendo I uma observação de I^* . Além disso, em I^* , cada região apresenta pixels que compartilham propriedades de homogeneidade em comum:

- No interior de qualquer região, dado qualquer canal de cor $\in \{R, G, B\}$, os pixels estatísticos apresentam mesmo valor esperado para esse canal de cor.
- As esperanças de regiões estatísticas adjacentes são diferentes para pelo menos um canal de cor $\in \{R, G, B\}$.

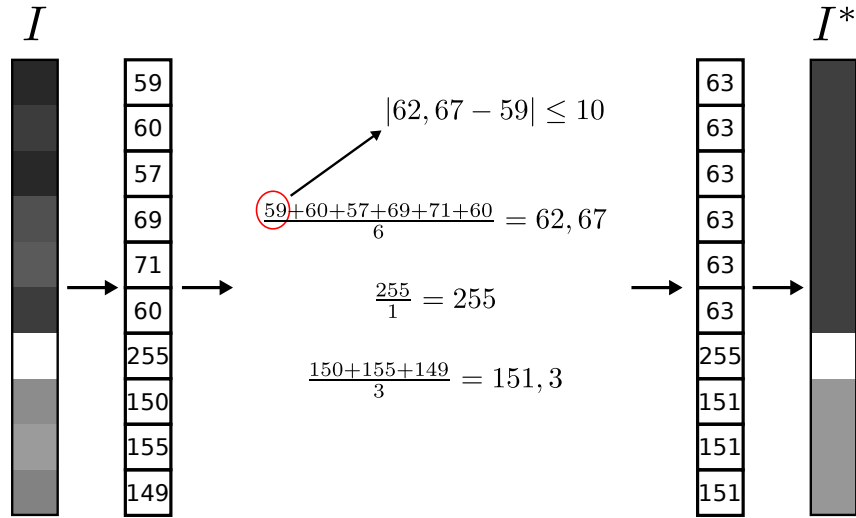
A decisão de fusão de duas regiões é tomada ao se comparar o valor esperado das regiões por meio de um critério de agrupamento.

Por exemplo, dado um conjunto de 10 pixels consecutivos (formando uma linha na imagem) cujo valor de suas respectivas intensidades é: 59, 60, 57, 69, 71, 60, 255, 150, 155 e 149, pode-se encontrar um conjunto de $N \leq 10$ regiões estatísticas formadas a partir desses pontos. Considerando o critério de agrupamento como uma verificação de limiar que estabelece que duas regiões A e B devem ser agrupadas caso a intensidade de todos os pixels de A e B estejam a uma “distância máxima” de $\tau = 10$ da intensidade média de $A \cup B$, pode-se conferir na Figura 59 as regiões estatísticas encontradas.

No exemplo da Figura 59 o conjunto de pontos inicial representa os pixels de I , e o conjunto final obtido pelo processo, I^* .

O limiar do algoritmo SRM é calculado dinamicamente pela intensidade média de cada região comparada e por um fator Q , o único parâmetro do algoritmo. Esse fator altera a complexidade estatística da cena, pois a distribuição de cada pixel de I^* é formada

Figura 59 – Exemplo do funcionamento do SRM para um vetor de 10 pixels, considerando que o critério de agrupamento é dado pela verificação de um limiar $\tau = 10$ em relação à média de cada região. Os vetores numéricos indicam a intensidade dos pixels do vetor de cores adjacente.



Fonte: Produção do próprio autor.

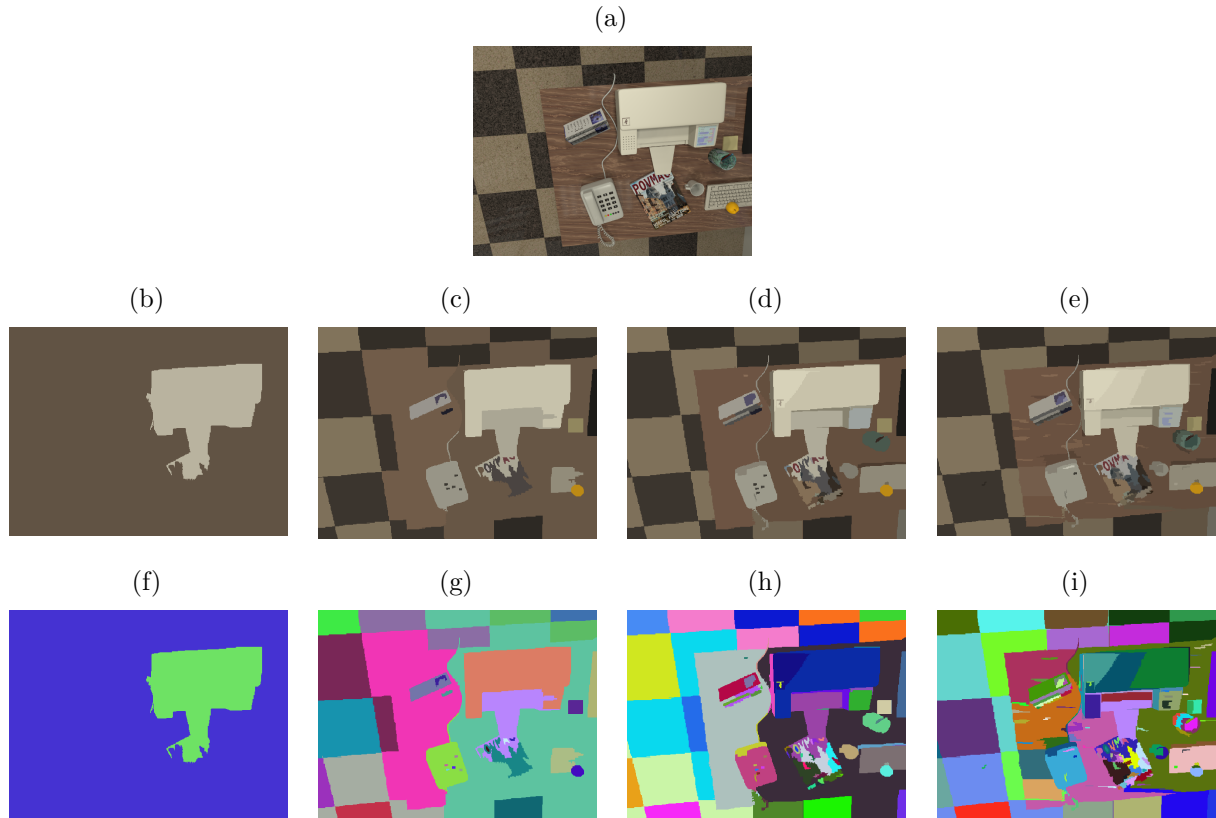
por uma combinação de Q variáveis aleatórias. Isso permite deixar a segmentação mais grosseira estatisticamente, levando a um menor número de regiões, consequentemente, a regiões maiores, como mostra a Figura 60.

A Figura 60 apresenta os resultados da segmentação por SRM da imagem I_r para diferentes valores de Q . Neste trabalho, a densificação da nuvem de pontos esparsa é realizada com a aproximação de planos em regiões homogêneas, portanto, é importante que a segmentação subdivida a imagem em regiões que apresentem pixels de um mesmo plano, ou pelo menos que sejam regiões de superfícies contínuas. O que intuitivamente leva à escolha de valores maiores de Q para uma segmentação menos grosseira, entretanto, quanto maior o valor de Q , mais subdividida fica I^* , facilitando o aparecimento de regiões pequenas contidas em regiões homogêneas. Por estarem no interior de regiões com pouca textura, há a possibilidade da região não conter pixels reconstruídos, o que não permitirá a estimação da superfície dessa região. Por isso, o valor escolhido para Q foi de 256.

O problema de subsegmentação apresentado na Seção 3.4.1, onde uma região do chão foi segmentada pelo MSER juntamente da mesa, também ocorre para o SRM.

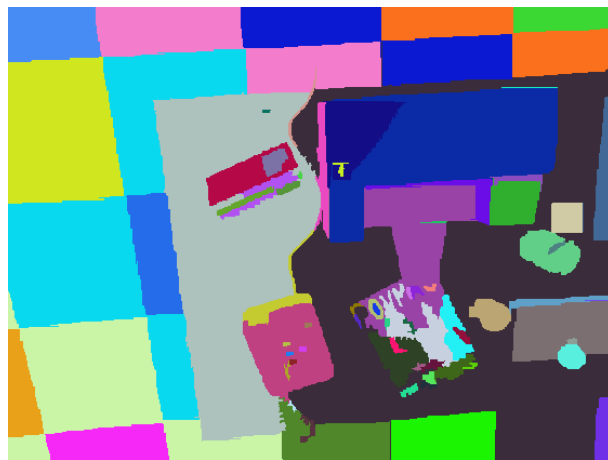
Para uma melhor visualização de cada região, a Figura 61 reapresenta o resultado da segmentação da imagem mostrada na Figura 60h para $Q = 256$, onde cada cor representa uma região diferente.

Figura 60 – A imagem RGB I_r (a) foi segmentada utilizando a técnica SRM, cujo parâmetro de configuração Q foi escolhido igual a 2 (b), 32 (c), 256 (d) e 1024 (e). O resultado da segmentação é melhor visualizado quando cada região corresponde a uma cor diferente, portanto, (f), (g), (h) e (i) apresentam esse resultado para as segmentações de (b), (c), (d) e (e), respectivamente.



Fonte: Produção do próprio autor.

Figura 61 – Resultado da segmentação da imagem mostrada na Figura 60a para $Q = 256$, onde cada cor representa uma região diferente.



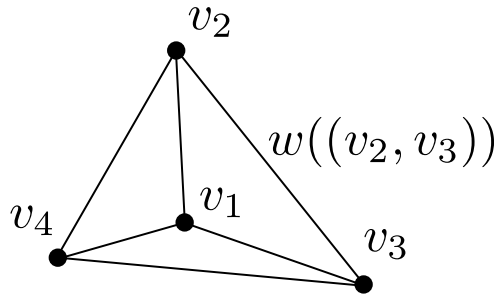
Fonte: Produção do próprio autor.

3.4.3 Superpixel

A técnica de segmentação de imagens Superpixel proposta por Felzenszwalb e Huttenlocher (2004) baseia-se no conceito de componentes conectados em um grafo.

A teoria dos grafos estuda as relações entre elementos de um conjunto (BIGGS; LLOYD; WILSON, 1976). Um grafo $G = (V, E)$ é formado por um conjunto V não vazio de elementos denominados vértices e um subconjunto E de pares não ordenados de V , chamados arestas. As arestas $(v_i, v_j) \in E$ indicam as possíveis ligações entre elementos v_i e v_j de um grafo, apresentando um peso $w((v_i, v_j))$ agregado, como mostra a Figura 62.

Figura 62 – Grafo formado por vértices $v_i \in V$ e arestas $(v_i, v_j) \in E$. $w((v_i, v_j))$ indica o peso da aresta (v_i, v_j) .



Fonte: Produção do próprio autor.

Para o caso de imagens, v_i e v_j são pixels e $w((v_i, v_j))$ é uma medida não negativa da dissimilaridade entre eles (e.g., a diferença absoluta da intensidade dos pixels).

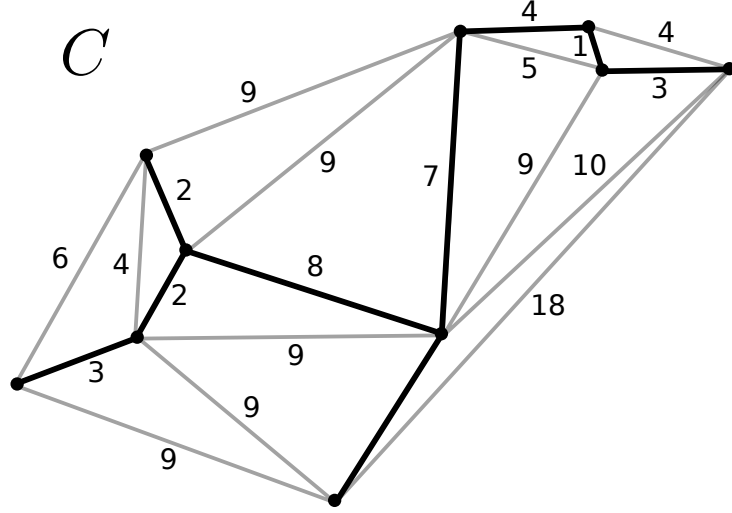
Na abordagem de Felzenszwalb e Huttenlocher (2004), uma imagem segmentada S é uma partição de V em componentes (regiões), tal que cada componente $C \in S$ corresponde a um componente conectado em um grafo $G' = (V, E')$, onde $E' \subseteq E$. Em outras palavras, qualquer segmentação é induzida por um subconjunto das arestas E .

A ideia é quantificar a dissimilaridade pelos pesos, onde as arestas de um componente devem possuir pesos relativamente pequenos, e arestas entre diferentes componentes devem apresentar pesos mais altos.

Cada componente C_i pode ser representado pelo subconjunto de E que conecte todos os vértices sem nenhum laço e com a menor soma total possível dos pesos das arestas, como apresenta a Figura 63. Esse arranjo resultante é chamado de **minimum spanning tree** ($MST(C_i, E)$) (KRUSKAL, 1956).

Para formar os componentes de S , deve-se, por meio de um processo iterativo, definir quando ou não juntar 2 componentes diferentes. O predicado D avalia isso, retornando verdadeiro quando deve-se juntar dois componentes C_1 e C_2 , ou falso, caso contrário. O predicado D é definido pela comparação entre as diferenças inter-componentes $Dif(C_1, C_2)$ e intra-componente de cada um ($Int(C_1)$ e $Int(C_2)$).

Figura 63 – Exemplo de *minimum spanning tree* para um componente C . As arestas que formam a *spanning tree* estão representadas de preto, e as demais do componente C de cinza. Tanto nesse exemplo quanto na técnica de Superpixels proposta por Felzenszwalb e Huttenlocher (2004), G é um grafo não direcional, ou seja, não existe um sentido único possível de se ir entre elementos.



Fonte: Riedl e Sefidgar (2014) (adaptado pelo autor).

A diferença intra-componente de C_i é dada pelo maior peso da *minimum spanning tree* desse componente, $MST(C_i, E)$, ou seja,

$$Int(C) = \max_{e \in MST(C_i, E)} w(e). \quad (3.50)$$

Já a diferença inter-componente entre C_1 e C_2 é definida com o menor peso do conjunto de arestas que conectam os dois componentes, portanto,

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w((v_i, v_j)). \quad (3.51)$$

Se não há arestas que conectem C_1 e C_2 , $Dif(C_1, C_2) = \infty$. Dessa maneira, o predicado D pode ser encontrado por

$$D(C_1, C_2) = \begin{cases} verdadeiro & \text{se } Dif(C_1, C_2) > MInt(C_1, C_2), \\ falso & \text{caso contrário} \end{cases}, \quad (3.52)$$

sendo

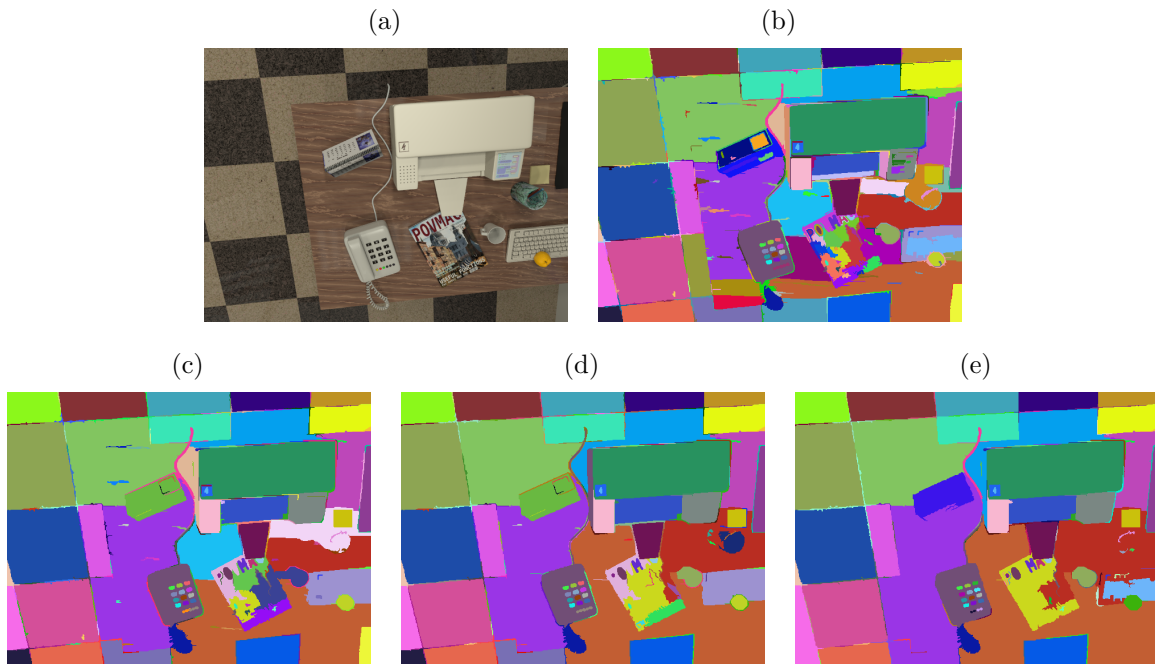
$$MInt(C_1, C_2) = \min\left(Int(C_1) + \frac{k}{|C_1|}, Int(C_2) + \frac{k}{|C_2|}\right), \quad (3.53)$$

onde $|C_1|$ e $|C_2|$ representam o tamanho (número de vértices) dos componentes C_1 e C_2 , respectivamente, e k um parâmetro do algoritmo. Para valores altos de k , a segmentação criará preferencialmente regiões maiores.

Dessa maneira, o algoritmo verifica para cada aresta $(v_i, v_j) \in E$ do grafo G , ou seja, para cada aresta formada a cada 2 pixels da imagem, se os componentes que contenham v_i e v_j atendem a Equação 3.52. Caso atenda, os dois componentes são agrupados, formando um novo componente que contém v_i e v_j . No início do processo, um filtro gaussiano de $\sigma = 0,8$ é utilizado para suavizar a imagem antes de calcular os pesos das arestas. Ao final do processo, S representa a imagem segmentada.

A Figura 64 apresenta os resultados da segmentação pela técnica Superpixels na imagem I_r para diferentes valores de k .

Figura 64 – A imagem RGB I_r (a) foi segmentada utilizando a técnica de Superpixels, cujo parâmetro de configuração k foi escolhido igual a 300 (b), 500 (c), 700 (d) e 1000 (e). No resultados da segmentação, cada região corresponde a uma cor diferente.



Fonte: Produção do próprio autor.

Como pode ser visto na Figura 64, valores altos de k levam a uma segmentação menos subdividida.

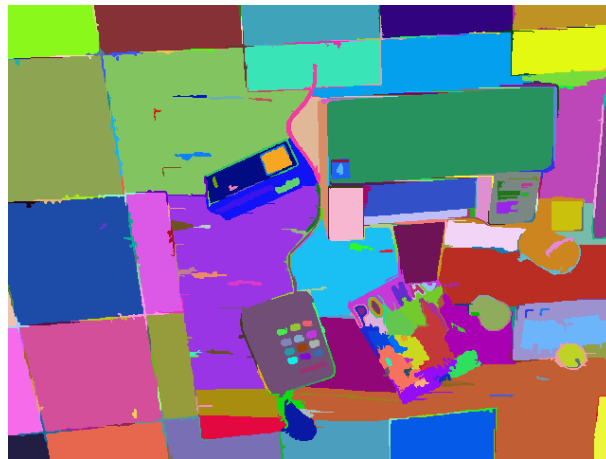
Pode-se observar também que, independente dos valores de k testados, a segmentação por Superpixel apresentou o mesmo problema de subsegmentação da região formada por uma área da mesa e uma do chão encontrados pelas técnicas MSER e SRM. Entretanto, todos os resultados da técnica de Superpixels apresentaram outros casos onde a mesa se

junta ao chão, sendo o resultado da Figura 64b ($k = 300$) o que apresentou menos casos que os demais ($k = 500, 700e1000$).

Felzenszwalb e Huttenlocher (2004) utiliza o valor de $k = 300$, referente ao resultado da Figura 64b. Para a comparação das técnicas na Seção 3.4.4, k também será considerado igual a 300, ou seja, o resultado mostrado na Figura 64b.

Para uma melhor visualização de cada região, a Figura 65 apresenta o resultado da segmentação da imagem mostrada na Figura 64b para $k = 300$, onde cada cor representa uma região diferente.

Figura 65 – Resultado da segmentação da imagem mostrada na Figura 64b para $k = 300$, onde cada cor representa uma região diferente.



Fonte: Produção do próprio autor.

3.4.4 Comparação entre as técnicas de segmentação de imagens

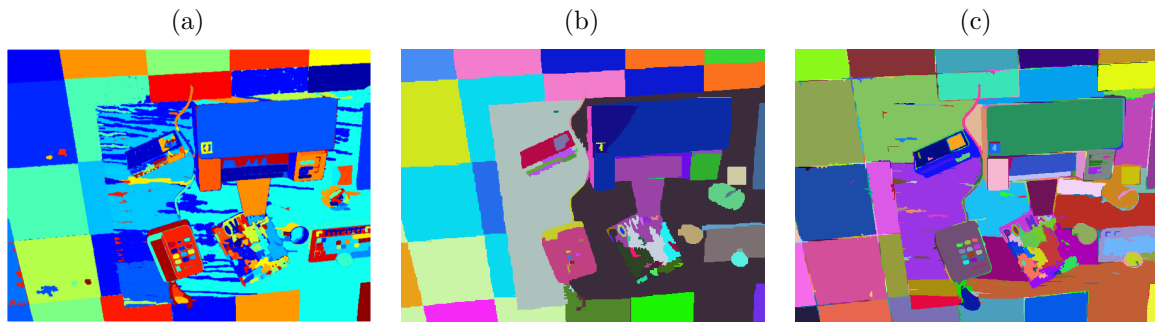
Os resultados dos algoritmos testados, mostrados nas Seções 3.4.1, 3.4.2 e 3.4.3, estão apresentados na Figura 66.

Como pode-se notar nas Figuras 66a e 66c, a imagem foi muito subdividida. Essa sobresegmentação pode prejudicar a reconstrução final, visto que fica mais fácil de aparecerem regiões pequenas contidas em regiões homogêneas. Por estarem no interior de regiões com pouca textura, há a possibilidade de que essas pequenas regiões não venham a ter pixels reconstruídos, o que não permitirá a estimação correta da superfície.

Dos três resultados, todos apresentaram problemas de subsegmentação, juntando áreas descontínuas, como o caso da mesa e do chão, sendo as técnicas do MSER e Superpixels as que mais apresentaram esse tipo de caso.

O objetivo nessa etapa é conseguir separar regiões que possuam características de cor similar. Entretanto, quando se considera que cada região homogênea é definida por um

Figura 66 – Resultado da segmentação da imagem apresentada na Figura 55a pelos métodos MSER (a), SRM (b) e Superpixels (c).



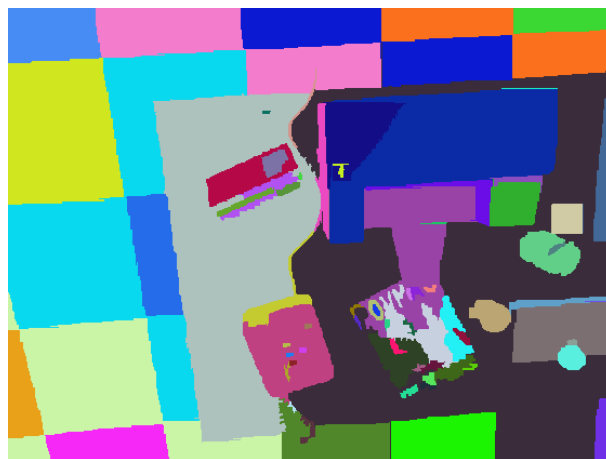
Fonte: Produção do próprio autor.

plano, quanto melhor segmentada em termos de regiões cujos pixels formem um mesmo plano, maiores serão as chances de se obter uma boa estimativa final de profundidade.

No caso deste trabalho, como o SRM apresentou na maioria das vezes uma segmentação da imagem em regiões homogêneas, sem criar muitas subregiões pequenas que pudessem prejudicar a reconstrução, essa foi a técnica escolhida para ser utilizada na etapa final de tratamento de regiões homogêneas e densificação da nuvem de pontos.

O algoritmo do SRM recebe a imagem filtrada por um filtro gaussiano 3×3 e retorna a matriz I_L contendo os *labels* de cada região. A Figura 67 apresenta as regiões segmentadas pelo método de SRM i.e., I_L .

Figura 67 – Matriz I_L contendo as regiões segmentadas, onde cada cor representa uma região diferente.



Fonte: Produção do próprio autor.

É importante ressaltar que uma técnica de segmentação de imagem em regiões homogêneas não tem a finalidade de encontrar áreas planares mas sim regiões similares.

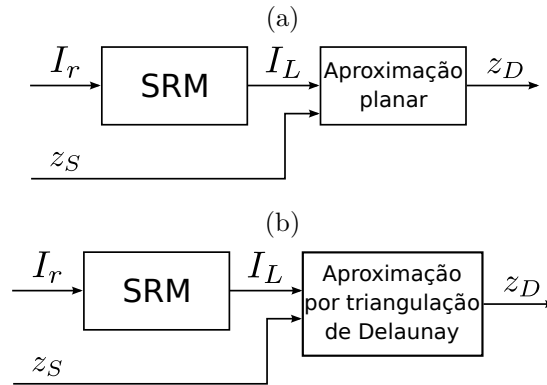
Dessa forma, nada impede que outros métodos possam ser usados para segmentar as imagens, dependendo das características e aspectos das cenas.

3.5 Reconstrução 3D densa

Os pixels reconstruídos, após as etapas de filtragem, representam uma nuvem de pontos esparsa. Para que a reconstrução 3D se torne densa, duas abordagens foram analisadas e propostas, aproximando-se regiões homogêneas por planos e por uma superfície a partir da triangulação de Delaunay. Nos dois casos, a imagem de referência é segmentada em regiões homogêneas de cor similar utilizando a técnica SRM.

Nesta seção, serão explicadas as duas abordagens propostas para a densificação da nuvem de pontos obtida da etapa de reconstrução esparsa. Os esquemáticos resumidos de ambas abordagens podem ser conferidos na Figura 23, já apresentados na Seção 3.2.

Figura 68 – Esquemático resumido dos algoritmos de tratamento de regiões homogêneas e densificação da nuvem de pontos. (a) algoritmo que aproxima cada região segmentada por um plano. (b) algoritmo que aproxima uma superfície por meio da triangulação de Delaunay dentro de cada região.

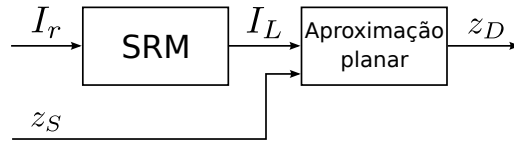


Fonte: Produção do próprio autor.

3.5.1 Primeira abordagem: cada região é um plano

Os pixels reconstruídos, após as etapas de fusão e filtragem, representam uma nuvem de pontos esparsa. Para que a reconstrução 3D torne-se densa, a imagem de referência I_r é segmentada em regiões homogêneas utilizando a técnica SRM. O SRM retorna uma matriz I_L , de mesmo tamanho que I_r , contendo os rótulos (*labels*) de cada região. A profundidade dos pixels de cada região de I_L é estimada por meio do plano obtido a partir da nuvem de pontos esparsa reconstruída e que está contida na região segmentada. No final desse processo, o resultado de estimativa densa de profundidade está representado por z_D , como pode ser visto na Figura 69.

Figura 69 – Esquemático resumido do algoritmo de tratamento de regiões homogêneas aproximando cada região segmentada por um plano.



Fonte: Produção do próprio autor.

Nessa abordagem, considera-se que cada região de I_L corresponde a uma área plana. Dessa forma, os pontos reconstruídos e não descartados pela etapa de filtragem podem ser utilizados para estimar o vetor normal ao plano correspondente a cada região ($\vec{n} = [n_u, n_v, n_z]^T$) e a distância do plano à câmera (d). Para isso, só são necessários 3 pontos não colineares (\mathbf{p}_1 , \mathbf{p}_2 e \mathbf{p}_3)⁹ por região, dado que a normal do plano \vec{n} é calculada como

$$\vec{n} = \vec{p}_{12} \times \vec{p}_{32}, \quad (3.54)$$

onde $\vec{p}_{12} = \mathbf{p}_1 - \mathbf{p}_2$ e $\vec{p}_{32} = \mathbf{p}_3 - \mathbf{p}_2$, e a distância d do plano à câmera é encontrada a partir de um dos pontos e da normal

$$d = \vec{n}^T \mathbf{p}_1. \quad (3.55)$$

Conhecidos \vec{n} e d , pode-se estimar a profundidade de todos os pixels (u, v) da região por meio da equação de plano

$$z = \frac{d - n_u u - n_v v}{n_z}. \quad (3.56)$$

Entretanto, os pontos reconstruídos de uma região nem sempre são coplanares, devido a ruídos na estimação, o que levará a um plano diferente dependendo da escolha dos três pontos (\mathbf{p}_1 , \mathbf{p}_2 e \mathbf{p}_3). Então como escolher os três pontos? E se um desses pontos for um *outlier* não descartado pelos procedimentos de filtragem?

Para resolver esses problemas, utiliza-se o algoritmo RANSAC (FISCHLER; BOLLES, 1981). Nesse algoritmo, um procedimento é repetido várias vezes (500 no caso deste trabalho) com o objetivo de encontrar o plano que melhor descreve os pontos de cada região. O Algoritmo 1 apresenta esse procedimento para a região de *label* k .

A cada iteração, 3 pontos (\mathbf{p}_1 , \mathbf{p}_2 e \mathbf{p}_3) são escolhidos aleatoriamente dentro dos possíveis pixels reconstruídos da região k ($\tilde{\mathbf{p}}_k = [u_k, v_k, z_k]^T$ ¹⁰). Posteriormente, \vec{n} e d são

⁹ o ponto \mathbf{p} é formado pelas coordenadas u e v do pixel, mais a sua profundidade z , $\mathbf{p} = [u, v, z]^T$

¹⁰ z_k é a profundidade, advinda de z_S , dos pixels reconstruídos $\tilde{\mathbf{p}}_k$ da região k . $z_k = z_S(v_k, u_k)$.

Algoritmo 1: ALGORITMO PARA ESTIMAÇÃO DA PROFUNDIDADE EM ÁREAS HOMOGÊNEAS

```

Entrada:  $\mathbf{u}_k, z_k$ 
Saída:  $\Pi_k, inmax$ 
início
   $\tilde{\mathbf{p}}_k = [u_k, v_k, z_k]^T$ 
   $inmax = 0$  /* número de inliers */
   $distin = \infty$  /* distância */
   $\Pi_k$  /* Plano ótimo */
  para  $n$  de 1 a 500 faça
     $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \in \tilde{\mathbf{p}}_k$  /* 3 pontos aleatórios */
     $\vec{p}_{12} = \mathbf{p}_1 - \mathbf{p}_2$ 
     $\vec{p}_{32} = \mathbf{p}_3 - \mathbf{p}_2$ 
     $\vec{n} = \vec{p}_{12} \times \vec{p}_{32}$ 
     $d = \vec{n}^T \mathbf{p}_1$ 
     $\Pi = [\vec{n}^T, d]^T$ 
     $z_p = \frac{\Pi(4) - \Pi(1)u_k - \Pi(2)v_k}{\Pi(3)}$ 
     $dist = |z_k - z_p|$ 
     $inliers = dist \leq 1,5 \text{ cm}$  /* vetor booleano marcando 1 para os
      inliers */
     $n\_in = \sum inliers$  /* número de inliers */
     $condição = (n\_in > inmax) \cup ((n\_in == inmax) \cap (dist(inliers) < distin))$ 
    se  $condição == TRUE$  então
       $inmax = n\_in$ 
       $distin = \text{sum}(dist(inliers))$ 
       $\Pi_k = \Pi$ 
    fim
  fim
retorna  $\Pi_k, inmax$ 
fim

```

calculados a partir desses três pontos. Com a Equação 3.56, pode-se descobrir a distância de cada ponto reconstruído da região k para este plano. Os pontos dados como *inliers* dessa hipótese de plano são aqueles que possuem distância menor que 1,5 cm. O plano $\Pi_k = [\vec{n}^T, d]$ com o maior número de *inliers* será retornado pelo algoritmo.

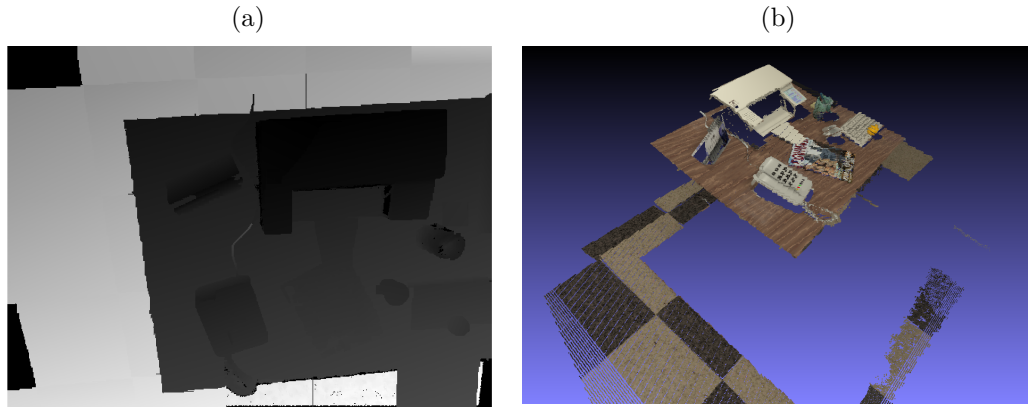
Encontrado o plano Π_k , o próximo passo é calcular a profundidade dos pixels pertencentes a cada região k por meio da relação

$$z_D(v_k, u_k) = \frac{\Pi_k(4) - \Pi_k(1)u_k - \Pi_k(2)v_k}{\Pi_k(3)}. \quad (3.57)$$

Porém, somente as regiões cujo número de *inliers* ($inmax$) seja de pelo menos 65% do número total de pixels reconstruídos destas regiões (\dot{u}_k) serão aproximadas por um plano. Isso ajuda a evitar que regiões não planas sejam aproximadas grosseiramente.

O mapa de profundidade e a nuvem de pontos resultante do processo de aproximação planar para o resultado apresentado na Figura 51 pode ser conferido na Figura 70.

Figura 70 – Mapa de profundidade (a) e nuvem de pontos (b) resultante do processo de aproximação planar para o resultado apresentado na Figura 51. O erro médio absoluto em relação ao *ground truth* foi de 3,54 cm contra 1,84 cm, sendo o percentual total de pixels reconstruídos de 96 % contra 60 %.



Fonte: Produção do próprio autor.

3.5.2 Segunda abordagem: cada região é uma superfície

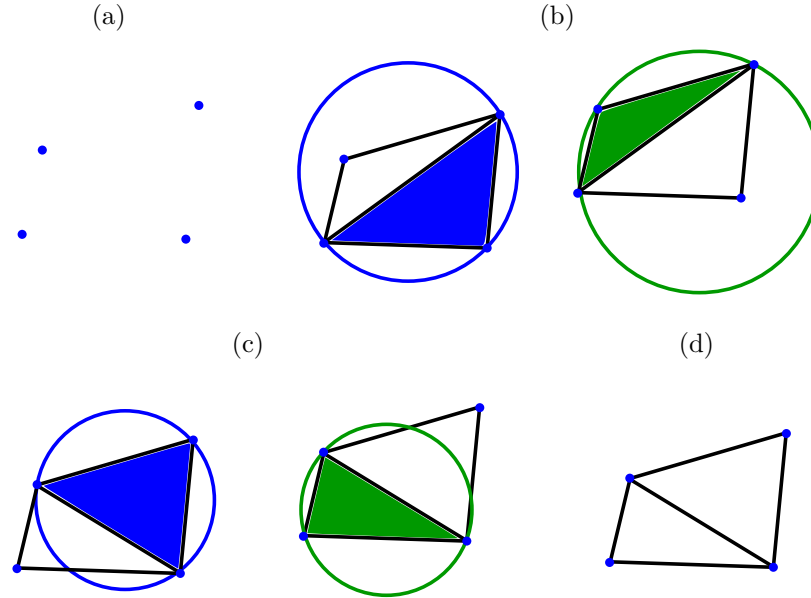
A Seção 3.5.1 apresentou uma abordagem onde as regiões homogêneas são aproximadas por planos. Entretanto, há a possibilidade do SRM segmentar junto regiões contínuas não planares ou que apresentam descontinuidades, como o caso já comentado da região que junta uma área do chão com a mesa na Figura 61. Para esses casos, a aproximação de planos pode resultar em uma estimação grosseira.

Geiger, Roser e Urtasun (2010) propõem um sistema rápido de *matching* chamado ELAS (*Efficient Large-Scale Stereo Matching*), onde pontos confiáveis que foram reconstruídos são utilizados para se estimar os demais. Para isso, considera-se que esses pontos confiáveis são vértices de triângulos, sendo que cada triângulo é um plano formado por três desses pontos. Essa consideração restringe a estratégia de aproximação de planos para uma área delimitada por apenas três pontos. Dessa forma, é possível gerar tanto planos em regiões homogêneas, quanto superfícies contínuas ou saltos descontínuos de profundidade. A técnica utilizada para encontrar esses triângulos é a **triangulação de Delaunay** (DELAUNAY, 1934).

A triangulação de Delaunay forma os triângulos de maneira que o círculo formado pelos vértices de um triângulo não contenha nenhum ponto em seu interior, como mostra a Figura 71.

Esse processo é realizado nos pixels reconstruídos na etapa de reconstrução 3D esparsa. Porém, a triangulação de Delaunay não é feita nos pontos tridimensionais, e sim no espaço 2D, como mostra a Figura 72, visto que se deseja encontrar triângulos que preencham os buracos entre pixels, adicionando posteriormente o conhecimento da

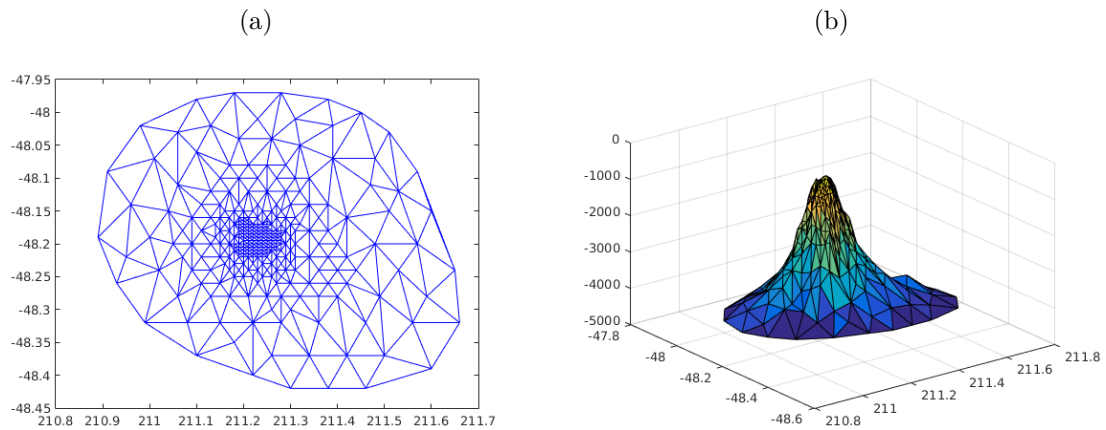
Figura 71 – (a) Pontos a serem transformados em triângulos pela triangulação de Delaunay. (b) Conjunto de triângulos que não formam uma triangulação de Delaunay. (c) Conjunto de triângulos que formam uma triangulação de Delaunay. (d) Triângulos resultantes do processo de triangulação de Delaunay.



Fonte: Produção do próprio autor.

profundidade para a geração dos planos.

Figura 72 – (a) Triangulação de Delaunay realizada em um conjunto de pontos e (b) o resultado da superfície ao adicionar o conhecimento da profundidade de cada um dos vértices dos triângulos.

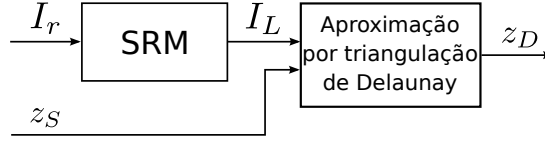


Fonte: Produção do próprio autor.

Com os triângulos encontrados pela triangulação de Delaunay e a matriz de profundidade z_S , pode-se estimar a profundidade dos pixels não reconstruídos. O esquemático

desse processo pode ser visto na Figura 73.

Figura 73 – Esquemático resumido do algoritmo de tratamento de regiões homogêneas aproximando cada região segmentada por uma superfície por meio da triangulação de Delaunay.



Fonte: Produção do próprio autor.

Entretanto, a triangulação de Delaunay é sensível a *outliers*. Mesmo que muitos *outliers* tenham sido filtrados durante a etapa de reconstrução 3D esparsa, ainda pode haver algum ponto mal reconstruído. Caso haja algum, se seu valor estimado de profundidade for muito diferente dos demais vizinhos, a estimativa poderá acumular muito erro. Isso acontece, pois os triângulos criados que tenham esse ponto como vértice representarão planos íngremes, o que propagará o erro de estimativa desse ponto para todos os pixels do interior desses triângulos.

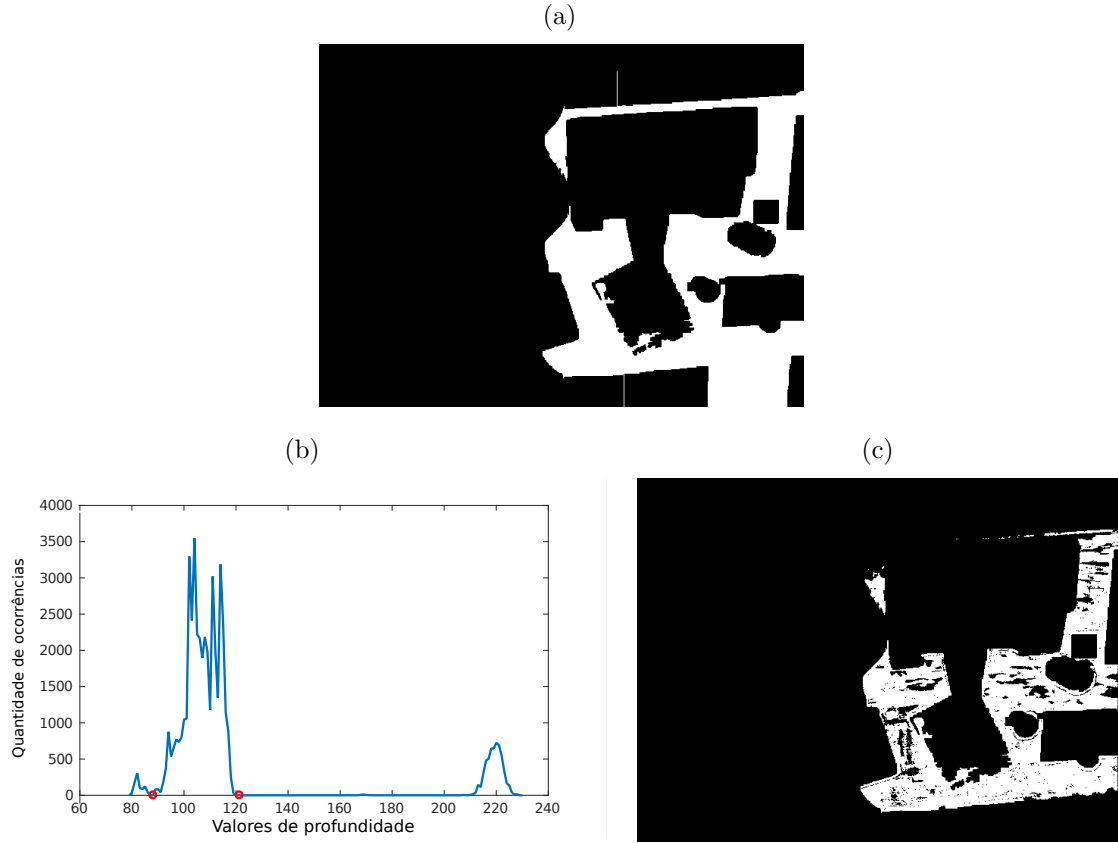
Geralmente, esses *outliers* aparecem como descontinuidades na nuvem de pontos, ou seja, entre esse pixel e o grupo de *inliers* não há uma continuidade de pontos tridimensionais. Portanto, analisar a continuidade dos valores de profundidade em uma certa região equivale a identificar no histograma de valores de profundidade dessa região qual o maior grupo contínuo de estimativas. Identificado o grupo de *inliers*, realiza-se para essa região a densificação com apenas esses pontos. A Figura 74 apresenta o histograma da região que junta uma área do chão com a mesa e os pontos dados como *inliers*, que serão utilizados para formar os triângulos de Delaunay.

Essa análise é realizada em cada região segmentada pelo SRM pois se fosse feita sobre a imagem inteira, não seria possível utilizar a filtragem de *outliers* por histograma de profundidade, dado que uma cena pode apresentar diversas regiões descontínuas, o que levaria a aproximação por triângulos de Delaunay para uma região apenas.

Além disso, todos os triângulos são criados no interior das regiões, não sendo possível formar arestas do lado de fora. Essa restrição do processo de criação dos triângulos de Delaunay é dada pela borda de cada região.

A equação de plano de cada triângulo que permite calcular a estimativa de profundidade dos pixels no interior de cada triângulo já foi apresentada na Seção 3.5.1. Sendo $\Pi_{\Delta} = [\vec{n}^T, d]$ os coeficientes do plano do triângulo Δ , a profundidade $z_D(v_{\Delta}, u_{\Delta})$ dos pixels (u_{Δ}, v_{Δ}) no interior deste triângulo é encontrada por

Figura 74 – Histograma dos valores de profundidade (b) da região destacada em branco (a). Os círculos vermelhos demarcam a região das ocorrências dadas como *inliers*, que determina os pontos utilizados para criar os triângulos (c), destacados em branco.

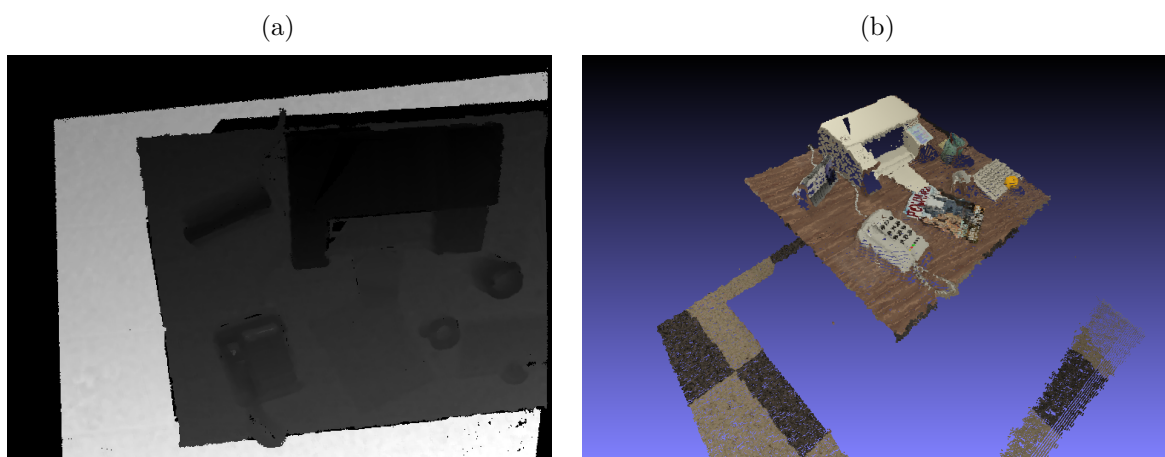


Fonte: Produção do próprio autor.

$$z_D(v_\Delta, u_\Delta) = \frac{\Pi_\Delta(4) - \Pi_\Delta(1)u_\Delta - \Pi_\Delta(2)v_\Delta}{\Pi_\Delta(3)}. \quad (3.58)$$

O mapa de profundidade e a nuvem de pontos resultante do processo de aproximação planar para o resultado apresentado na Figura 51 pode ser conferido na Figura 75.

Figura 75 – Mapa de profundidade (a) e nuvem de pontos (b) resultante do processo de aproximação planar para o resultado apresentado na Figura 51. O erro médio absoluto em relação ao *ground truth* foi de 1,65 cm contra 1,84 cm, sendo o percentual total de pixels reconstruídos de 77 % contra 60 %.



Fonte: Produção do próprio autor.

4 Experimentos e resultados

Para desenvolver e validar o método proposto, foram utilizadas imagens sintéticas foto-realistas geradas por um *software* chamado POVRay, disponibilizadas por Handa et al. (2012). Esse banco de dados também fornece a matriz de parâmetros intrínsecos K , e extrínsecos T_{wc} da câmera, além do *ground truth* da profundidade de todas as imagens.

As imagens apresentam, sobre uma mesa de madeira, uma impressora, um telefone, um calendário, uma revista, um copo, uma xícara, um teclado, uma caixa e uma laranja. Essa mesa se encontra em um escritório com chão quadriculado. Ao todo são utilizadas 10 imagens I_m que possuem certa sobreposição com uma imagem de referência I_r (Figura 76). Todas as imagens pertencem ao espaço de cor RGB.

Figura 76 – Imagem sintética foto-realista utilizada para os experimentos disponibilizada por Handa et al. (2012). Imagem de referência do conjunto de imagens.



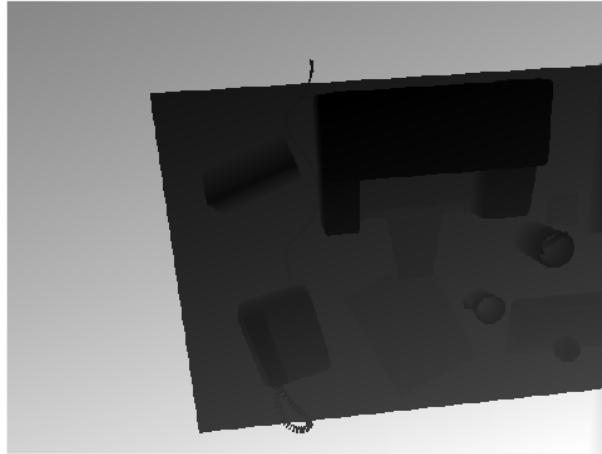
Fonte: Handa et al. (2012).

A profundidade real da cena (*ground truth*) para os pixels da Figura 76 pode ser visto na Figura 77.

O algoritmo foi analisado em termos de *recall* (porcentagem de pixels reconstruídos) e erro absoluto em relação ao *ground truth* para os pixels reconstruídos. Ao final, parte dos resultados obtidos com o método proposto foi comparada com os do DTAM, que apresenta um bom compromisso entre velocidade e qualidade, além de ser amplamente citado na literatura.

O DTAM estima, primeiramente, a profundidade de todos os pixels, e depois refina esse resultado por meio de um processo iterativo de minimização. Dessa forma, seus resultados foram separados em: estimativa inicial (Rec. inicial) e o resultado após o

Figura 77 – Mapa de profundidade do *ground truth* da imagem de referência.



Fonte: Handa et al. (2012).

processo de minimização (Rec. final). Para os testes referentes ao DTAM, foram usadas 30 imagens e 32 níveis discretos de profundidade.

Os resultados do trabalho proposto foram separados quanto a métrica utilizada (SAD, SSD e FNCC) e subdivididos de acordo com a opção de verificação da consistência esquerda-direita (*LRC*) e de pirâmide gaussiana (*Pyr*), sendo que essa pirâmide terá 2 níveis. Esses resultados serão apresentados para as etapas de reconstrução esparsa, reconstrução 3D com aproximação por planos (Planos) e reconstrução 3D com aproximação por superfície utilizando triangulação de Delaunay (Delaunay).

O algoritmo de reconstrução 3D esparsa proposto foi desenvolvido em *C++* utilizando as bibliotecas computacionais OpenCV (BRADSKI, 2000) (biblioteca de visão computacional) e Armadillo (SANDERSON; CURTIN, 2016) (biblioteca de álgebra linear). Além disso, o algoritmo foi testado em um microcomputador com processador Intel i5 – 4590 quad-core 3.30 *GHz* 64 bits, equipado com 8 *GB* de memória RAM. As etapas de tratamento de regiões homogêneas por aproximação de planos e por aproximação de superfície utilizando triangulação de Delaunay foram implementadas no Matlab®.

4.1 Reconstrução 3D esparsa

Nesta seção, serão apresentados os resultados referentes à etapa de reconstrução 3D esparsa. Vale comentar que os resultados das etapas intermediárias da metodologia proposta responsáveis pelo refinamento da reconstrução (suspixels e refinamento local por parábola) foram mostrados no decorrer do Capítulo 3.

Para comparar as métricas em relação ao tempo gasto por cada uma, e para dar uma ideia do tempo de processamento do algoritmo, algumas medidas foram realizadas:

- t_m : tempo total gasto para realizar o processo de *matching* dos 10 pares de imagens.
- t_T : tempo total gasto para realizar a etapa de fusão das estimativas de profundidade (etapa de consistência temporal).
- t_S : tempo total gasto para realizar a etapa de filtragem espacial com o filtro de mediana (etapa de consistência espacial).

Como o tempo depende do computador utilizado, este resultado visa unicamente comparar proporcionalmente o custo entre as métricas. É importante lembrar que a medida de tempo de processamento também depende dos processos do sistema operacional, portanto, ao invés de se utilizar as medidas de t_m , t_T e t_S de uma única execução do algoritmo, foram realizadas 10 medidas (10 execuções), utilizando-se então a média desses resultados

$$\bar{t}_m = \frac{1}{10} \sum_{i=1}^{10} t_m(i), \quad (4.1)$$

$$\bar{t}_T = \frac{1}{10} \sum_{i=1}^{10} t_T(i), \quad (4.2)$$

$$\bar{t}_S = \frac{1}{10} \sum_{i=1}^{10} t_S(i), \quad (4.3)$$

onde $t_m(i)$, $t_T(i)$ e $t_S(i)$ são os respectivos valores de t_m , t_T e t_S na i -ésima execução do algoritmo. \bar{t}_m , \bar{t}_T e \bar{t}_S são os tempos médios das 10 medidas realizadas. O tempo total é calculado como

$$\text{Tempo Total} = \bar{t}_m + \bar{t}_T + \bar{t}_S. \quad (4.4)$$

Além disso, como o processo de *matching* é independente para cada par de imagens, pode-se realizar esse processo de forma paralela. Para isso, utilizou-se um sistema de “workers” (trabalhadores) para agilizar o processo. Nesse sistema, a etapa de *matching* é considerada um tipo “trabalho” a ser realizado pelos trabalhadores. Como são 10 pares de imagens, os trabalhadores tem no total 10 trabalhos a serem resolvidos. No caso em questão, cada *worker* corresponde a um núcleo do computador. Dessa forma, cada *worker* retira da fila de 10 trabalhos um deles, processa, retorna a profundidade resultante do *matching* daquele par de imagens e verifica a existência de mais trabalhos na fila de trabalhos. Caso ainda haja algum par de imagens a ser processado, o trabalhador realiza essa sequência novamente.

Neste trabalho, a etapa de *matching* foi processada por 3 *workers*, ou seja, de forma paralela por 3 dos 4 núcleos do computador. Entretanto, esse sistema pode ser estendido

Tabela 1 – Resultados da reconstrução esparsa para as métricas SAD, SSD e FNCC.

SAD									
#	Técnica*		Erro (cm)					Recall (%)	Tempo Total (s)
	LRC	Pyr	Média	Mediana	Desvio Padrão	Mínimo**	Máximo		
1		✓	36,47	0,87	155,27	0	894,86	61,7	8,67
2	✓	✓	2,76	0,64	12,37	0	810,52	49,0	14,60
3			26,04	0,71	135,3	0	895,91	67,6	41,68
4	✓		2,19	0,64	9,67	0	835,31	62,5	82,07
SSD									
#	Técnica*		Erro (cm)					Recall (%)	Tempo Total (s)
	LRC	Pyr	Média	Mediana	Desvio Padrão	Mínimo**	Máximo		
1		✓	37,97	0,92	157,74	0	894,86	61,5	10,22
2	✓	✓	2,92	0,68	12,53	0	197,17	48,6	17,64
3			26,63	0,73	136,63	0	839,62	67,6	55,24
4	✓		2,32	0,67	10,05	0	835,31	62,4	105,90
FNCC									
#	Técnica*		Erro (cm)					Recall (%)	Tempo Total (s)
	LRC	Pyr	Média	Mediana	Desvio Padrão	Mínimo**	Máximo		
1		✓	6,37	0,82	42,25	0	838,69	58,6	14,34
2	✓	✓	2,19	0,64	10,12	0	152,33	46,6	25,09
3			6,98	0,66	58,40	0	839,60	65,8	88,95
4	✓		1,84	0,64	7,88	0	136,19	61,0	159,37

* Técnicas utilizadas no algoritmo: consistência esquerda-direita (LRC) e pirâmide gaussiana de 2 níveis (Pyr).

** Valores nulos de erro são, na verdade, valores menores que 10^{-5} .

para quantos núcleos forem necessários, inclusive de outras máquinas, por meio de uma conexão de internet. Foi possível realizar a etapa de *matching* dessa maneira pelo fato de todo o processo ser implementado como *single-core*, ou seja, utilizando apenas um núcleo em toda a execução.

No total, foram realizados quatro experimento por métrica: não verificando LRC e utilizando pirâmide, verificando LRC e utilizando pirâmide, não verificando LRC e não utilizando pirâmide, e verificando LRC e não utilizando pirâmide, identificados pelos números 1, 2, 3 e 4, respectivamente. Os resultados desses experimentos para a etapa de reconstrução 3D esparsa para as três métricas estão apresentados na Tabela 1.

A Tabela 1 apresenta os valores estatísticos de erro absoluto em centímetros (média, mediana, etc.), de *recall* em porcentagem de pixels reconstruídos da imagem pelo número de pixels total, e de tempo total gasto de processamento em segundos.

Como pode-se ver na Tabela 1, o SAD foi a métrica que atingiu os valores de tempo mais baixos em cada experimento, sendo o caso em que se utiliza pirâmide e não se verifica a consistência esquerda-direita o que apresenta o menor deles. Isso pode ser explicado pelo SAD ser a métrica menos custosa computacionalmente das três. Com a pirâmide, a quantidade de contas realizadas diminui ainda mais, como explicado no Capítulo 3. A não verificação do LRC também contribui para a velocidade de reconstrução, visto que esta exige que o processo de *matching* seja realizado duas vezes, praticamente dobrando o tempo de execução do algoritmo. Entretanto, verificar o LRC permitiu um erro absoluto menor para todas as métricas testadas.

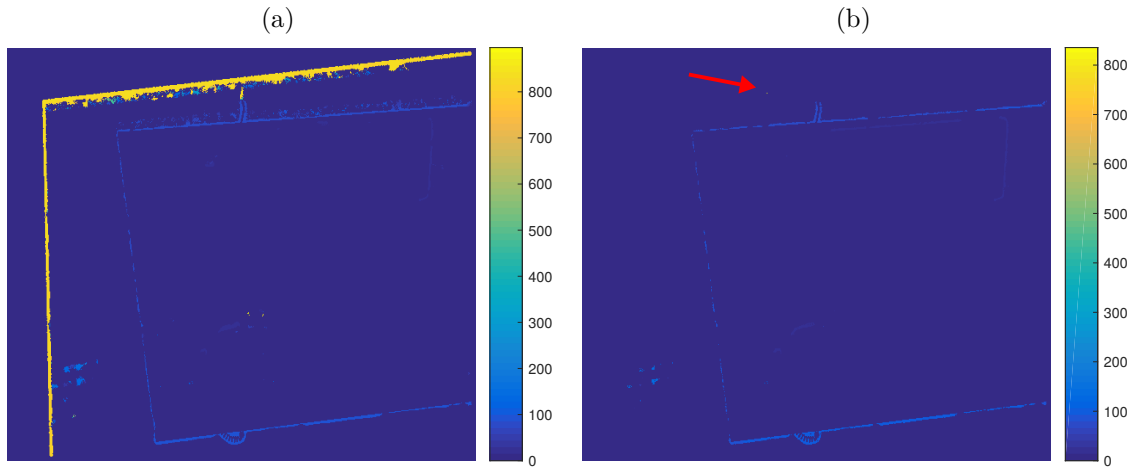
A utilização de pirâmide não implica em diminuição do erro, e sim na diminuição do tempo de processamento. Tempos esses cerca de 82,5 % menores em relação aos seus experimentos correspondentes sem a utilização de pirâmide. Isso acontece pela diminuição da complexidade a cada nível a mais adicionado na pirâmide, como comentado no Capítulo 3. Todavia, ao se considerar que a linha epipolar do nível inferior da pirâmide tem um tamanho de três pixels, o problema de busca pelo correspondente se restringe àqueles três pixels apenas, diminuindo o tempo de processamento total da reconstrução, mas possibilitando a convergência equivocada do resultado de estimação para uma determinada região. Dessa forma, uma má localização do correspondente em níveis superiores, pode impossibilitar a correspondência correta em níveis inferiores.

Além disso, pode-se notar que o *recall* sempre diminui quando se utiliza pirâmide. Isso acontece porque os pixels descartados, por serem considerados suspixels, cuja estimação de profundidade é duvidosa/suspeita em um nível superior da pirâmide, geram o descarte de 4 outros pixels no próximo nível. Assim, em uma pirâmide de 3 níveis, um pixel descartado no nível superior, levará a dezesseis pixels descartados no nível base.

A Figura 78 mostra o resultado do erro absoluto para os experimentos SAD4 e SAD3 do algoritmo sem a utilização de pirâmide com (Figura 78b) e sem (Figura 78a) a verificação da consistência esquerda-direita, respectivamente. Como pode-se notar, no resultado obtido quando não se verifica o LRC (SAD3), o erro absoluto na ordem de 800 *cm* está concentrado formando um entorno bem definido. Esse entorno é formado pelo limite da área de sobreposição entre as imagens com a imagem de referência. O erro absoluto alto dessa região pode ser justificado pelo fato de não ser possível encontrar a localização correta dos pixels desse entorno nas outras imagens, dado que os correspondentes corretos desses pixels estão fora dos limites das outras imagens. Caso aconteça de uma parte da linha epipolar desses pontos existir dentro das outras imagens, o pixel dado como melhor correspondente será escolhido na etapa de *matching*. Como o pixel correspondente correto não está nas imagens, o melhor correspondente levará a uma má estimação da profundidade, acarretando esse tipo de erro.

Quando se verifica a consistência esquerda-direita, a maioria desses pontos mal

Figura 78 – Erro absoluto do experimento do algoritmo para a métrica SAD sem a utilização de pirâmide sem (a) e com (b) verificação da consistência esquerda-direita (SAD3 e SAD4, respectivamente). A barra à direita da imagem indica a cor correspondente aos valores de erro absoluto em relação a *ground truth* em *cm*. Um pixel mal estimado em (b) que não foi descartado está indicado por uma seta vermelha.



Fonte: Produção do próprio autor.

estimados é descartada. Pode acontecer de um ou outro ponto sobrar, levando ao caso apresentado na Figura 78b, indicado pela seta vermelha. Esse pixel mal estimado que não foi descartado é o responsável pelo erro máximo alto (835,31 *cm*), mas não é o suficiente para influenciar tanto na média, no desvio padrão e na mediana do erro absoluto, que permanecem baixos para o resultado da Figura 78b na Tabela 1 (SAD4). Os outros casos que elevam o valor do erro médio do resultado da Figura 78b são provenientes dos pixels das bordas de regiões onde há descontinuidade, como é o caso das bordas das regiões que sofrem oclusão da tampa da impressora e da mesa. Nesses casos, os pixels da mesa e do chão, respectivamente, são reconstruídos incorretamente como parte da impressora e da mesa, como explicado no Capítulo 3.

Mesmo com os pixels mal estimados não descartados pelo algoritmo, a verificação da consistência esquerda-direita se mostrou importante para a retirada de *outliers*. Sem essa etapa, o *recall* é maior, entretanto, essa quantidade maior de pixels reconstruídos se deve, muitas vezes, a pontos mal estimados, como visto na Figura 78. Isso também pode ser observado na Tabela 1: quando se adiciona a verificação LRC ao método, o erro médio absoluto do resultado diminui de 2,9 a 13 vezes.

Vale comentar que os valores de *recall* apresentados na Tabela 1 não são tão baixos quanto se pode parecer. Reconstruir a imagem completamente só é possível quando a sobreposição das 10 imagens sobre a referência contempla todos os seus pixels. Entretanto, o conjunto de imagens utilizado não permite isso, sendo a sobreposição máxima entre as

imagens de 82,5 % da imagem de referência, como mostra a Figura 79.

Figura 79 – Sobreposição máxima entre as imagens e a imagem de referência (cor branca).



Fonte: Produção do próprio autor.

Dessa maneira, o *recall* máximo possível na etapa de reconstrução 3D esparsa é de 82,5 %, ou seja, o máximo *recall* obtido nos resultados (67,6 %) corresponde a 81 % desse valor.

Pode-se perceber também, que os resultados do SAD e do SSD foram muito parecidos. Isso se deve ao fato de que ambas métricas são calculadas a partir da soma do valor não negativo do erro. A diferença é que o SSD utiliza o erro quadrático, penalizando, dessa forma, os pixels da janela que apresentam um erro maior. Além disso, calcular o quadrado de um número é mais custoso computacionalmente que encontrar o módulo dele. Isso se reflete no tempo de execução do algoritmo, apresentado na Tabela 1, sendo o tempo de processamento do SSD maior que o do SAD para todos os experimentos. Mesmo assim, todos os resultados do SAD foram melhores que o SSD, apresentando erro e tempo menor ou igual e *recall* maior ou igual que o SSD. Por isso, o SSD não será comparado nas etapas de tratamento de regiões homogêneas com as métricas SAD e FNCC.

O algoritmo de correlação cruzada normalizada utilizado nos experimentos dessa seção é uma versão otimizada (FNCC) do NCC, como comentado no Capítulo 3. Um teste foi realizado para os experimentos da Tabela 1 utilizando-se o cálculo da métrica NCC não otimizada para a comparação do tempo de execução com o FNCC. Como o FNCC não altera o valor encontrado pelo NCC, os resultados desse teste foram semelhantes aos da Tabela 1 com exceção do tempo de processamento, que foi, em média, 42,5 % mais lento que o FNCC, justificando, portanto, o uso dessa métrica alterada.

O tempo de processamento depende dos 10 resultados de *matching* e das etapas de consistência temporal e espacial, como explicado no início dessa seção, sendo a etapa de *matching* a mais custosa das três. As duas últimas independem da métrica utilizada,

alterando seu tempo de execução de acordo com o *recall*, visto que quanto mais pontos foram reconstruídos, maior o número de pontos da imagem que terão uma estimativa válida para essas etapas, levando a um maior número de pontos fundidos e filtrados. Como a etapa de *matching* pode ser executada em paralelo, o tempo dessa etapa pode diminuir com a utilização de mais núcleos do processador (*workers*). Na melhor situação, 10 *workers* estão disponíveis para o sistema de reconstrução 3D, ou seja, 10 núcleos dedicados, podendo-se processar os 10 pares de imagens ao mesmo tempo.

Vale ressaltar que o código não foi otimizado, salve o NCC (otimizado para FNCC), nem foi utilizada uma GPU para os cálculos, sendo assim possível atingir resultados mais rápidos dependendo do sistema da aplicação. Além disso, de acordo com a aplicação que necessite de um serviço de reconstrução 3D, pode-se utilizar uma métrica mais rápida como o SAD, caso o tempo seja importante, ou uma métrica mais lenta como o FNCC caso o tempo não seja um empecilho. Por exemplo, em um ambiente com um sistema de câmeras fixas, onde deseja-se gerar um mapa tridimensional para o controle de robôs. Nesse caso, a reconstrução pode ser realizada pouco tempo antes do robô iniciar a sua tarefa.

Em resumo, os resultados de cada métrica que apresentaram o melhor compromisso entre precisão e velocidade foram os que verificaram o LRC e utilizaram pirâmide gaussiana (SAD2, SSD2 e FNCC2). Mesmo apresentando um *recall* menor, este resultado será melhorado nas etapas de densificação da nuvem de pontos. O mapa de profundidade e a nuvem de pontos dos melhores resultados de cada métrica da etapa de reconstrução 3D esparsa podem ser conferidos na Figura 80.

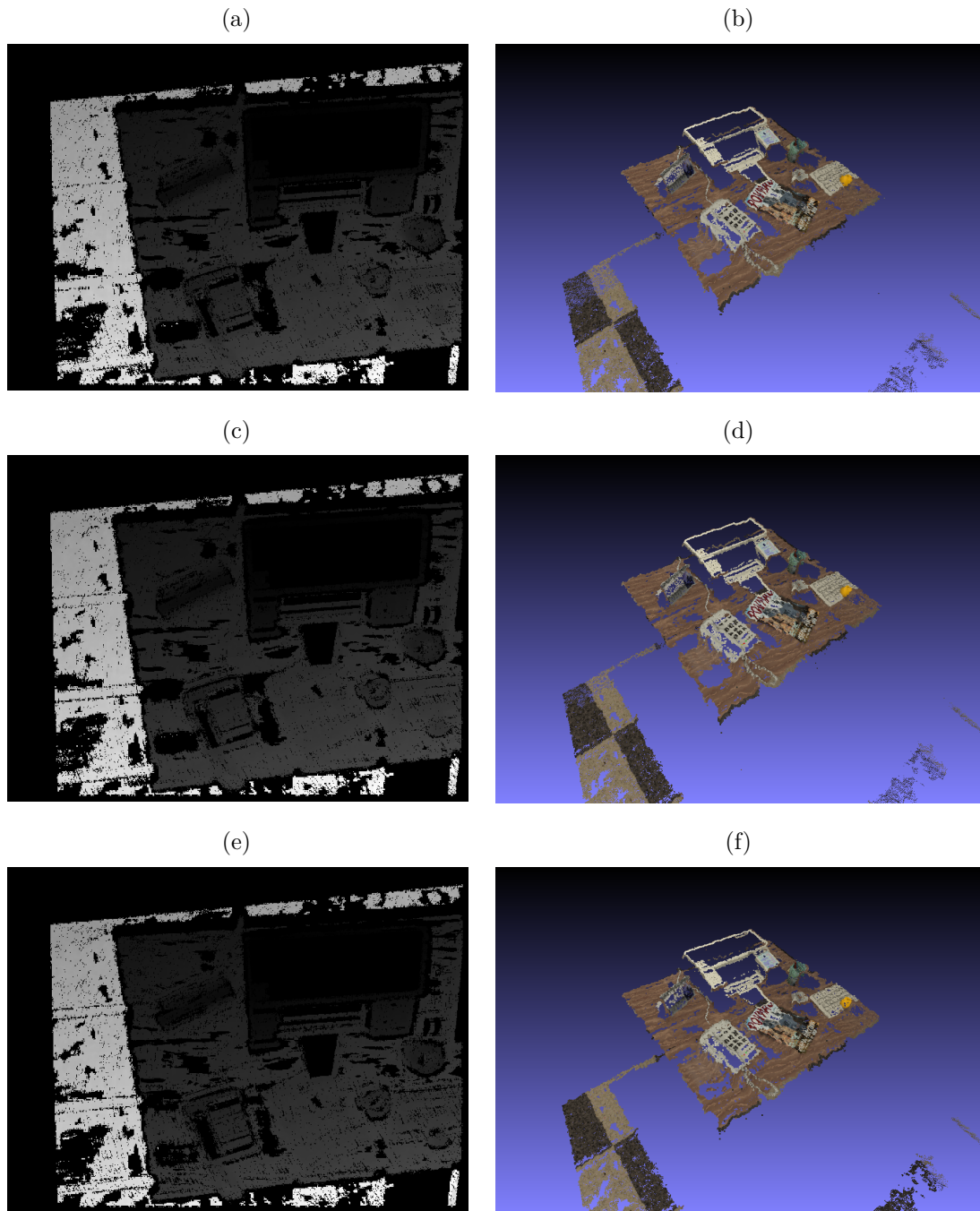
Como pode-se notar nos mapas de profundidade e na nuvem de pontos apresentados na Figura 80, todos os três resultados são bem parecidos visualmente, confirmando os dados mostrados na Tabela 1. Além disso, é possível notar os pixels mal reconstruídos responsáveis pelos erros absolutos máximos da Tabela 1 nas nuvens de pontos, presentes em sua maioria como pontos do chão que foram reconstruídos na altura da mesa.

Os resultados de reconstrução esparsa são densificados nos experimentos referentes à etapa de reconstrução 3D densa da Seção 4.2 e serão mostrados a seguir.

4.2 Reconstrução 3D densa

Nessa seção, serão apresentados os resultados referentes à etapa de reconstrução 3D densa com tratamento de regiões homogêneas para as duas abordagens explicitadas no Capítulo 3. Vale lembrar que a segmentação por SRM só é realizada na imagem de referência, pois se busca apenas a estimação de profundidade dos pixels de I_r . Além disso, seu resultado já foi apresentado no Capítulo 3, e por isso, não será comentado nessa seção.

Figura 80 – Mapas de profundidade dos melhores resultados da métricas SAD2 (a), SSD2 (c) e FNCC2 (e) e suas respectivas nuvens de pontos (b,d,f) para a etapa de reconstrução 3D esparsa.



Fonte: Produção do próprio autor.

Os experimentos dessa seção se resumem em aplicar o método de aproximação de regiões homogêneas por planos e por superfícies encontradas pela triangulação de Delaunay nos resultados obtidos dos experimentos realizados na Seção 4.1. Somente as métricas SAD e FNCC serão avaliadas nessa seção, como explicado na seção anterior. Os resultados desses experimentos para a etapa de reconstrução 3D densa para as duas métricas estão

Tabela 2 – Resultados da reconstrução 3D com aproximação por planos para as métricas SAD, SSD e FNCC.

SAD								
#	Técnica*		Erro (cm)					Recall (%)
	LRC	Pyr	Média	Mediana	Desvio Padrão	Mínimo**	Máximo	
1		✓	26,26	0,85	125,48	0	838,71	79,1
2	✓	✓	5,05	0,67	19,10	0	810,52	90,9
3			12,38	0,63	82,60	0	837,04	88,8
4	✓		3,73	0,64	15,94	0	835,31	92,8

FNCC								
#	Técnica*		Erro (cm)					Recall (%)
	LRC	Pyr	Média	Mediana	Desvio Padrão	Mínimo**	Máximo	
1		✓	6,01	0,71	30,65	0	834,90	81,8
2	✓	✓	4,82	0,66	18,31	0	101,95	94,7
3			5,44	0,66	36,56	0	834,68	90,6
4	✓		3,45	0,59	15,35	0	102,04	96,2

* Técnicas utilizadas no algoritmo: consistência esquerda-direita (LRC) e pirâmide gaussiana de 2 níveis (Pyr).

** Valores nulos de erro são, na verdade, valores menores que 10^{-5} .

apresentados nas Tabelas 2 e 3.

As Tabelas 2 e 3 apresentam os valores estatísticos de erro absoluto em centímetros (média, mediana, etc.) e de *recall* em porcentagem de pixels reconstruídos da imagem pelo número de pixels total. O tempo gasto de processamento na etapa de densificação não é calculado pelo desenvolvimento ter sido realizado em Matlab®. Dessa forma, para analisar qual métrica apresenta o melhor compromisso entre precisão e velocidade foram utilizados os dados de tempo mostrados na Tabela 1.

Como pode-se notar nas Tabelas 2 e 3, a aproximação por planos apresentou um maior *recall* para todos os experimentos. Isso se deve ao fato de que, a partir dos pontos de cada região segmentada pelo SRM, é possível estimar planos que formem regiões com partes fora da área de sobreposição das imagens, mesmo que esses pontos sejam poucos (só são necessários 3 pontos não colineares para se gerar um plano).

Já os resultados para aproximação por Delaunay apresentaram *recall* menor que o *recall* máximo possível de 82,5 %. Isso acontece pois a aproximação por Delaunay utiliza pontos já reconstruídos para estimar os que se localizam na região fechada delimitada por esses pontos, ou seja, não há como se encontrar os pixels fora da sobreposição das imagens por essa abordagem.

Pode-se observar também, que o *recall* dos melhores resultados da Seção 4.1 (SAD2

Tabela 3 – Resultados da reconstrução 3D com aproximação por superfície utilizando triangulação de Delaunay para as métricas SAD, SSD e FNCC.

SAD								
#	Técnica*		Erro (cm)					Recall (%)
	LRC	Pyr	Média	Mediana	Desvio Padrão	Mínimo**	Máximo	
1		✓	30,05	0,84	141,01	0	894,86	76,7
2	✓	✓	2,28	0,66	10,43	0	810,52	71,0
3			22,20	0,68	124,79	0	895,91	80,2
4	✓		1,98	0,64	8,81	0	835,31	77,1
FNCC								
#	Técnica*		Erro (cm)					Recall (%)
	LRC	Pyr	Média	Mediana	Desvio Padrão	Mínimo**	Máximo	
1		✓	5,38	0,84	37,90	0	838,69	73,8
2	✓	✓	1,89	0,70	8,56	0	152,33	68,8
3			6,01	0,65	53,48	0	839,60	78,7
4	✓		1,65	0,61	7,24	0	136,19	76,5

* Técnicas utilizadas no algoritmo: consistência esquerda-direita (LRC) e pirâmide gaussiana de 2 níveis (Pyr).

** Valores nulos de erro são, na verdade, valores menores que 10^{-5} .

e FNCC2) tiveram um aumento considerável: o *recall* do SAD aumentou de 49 % para 90,9 % (aproximação por planos) e 71 % (aproximação por Delaunay), e o *recall* do FNCC aumentou de 46,6 % para 94,7 % (aproximação por planos) e 68,8 % (aproximação por Delaunay).

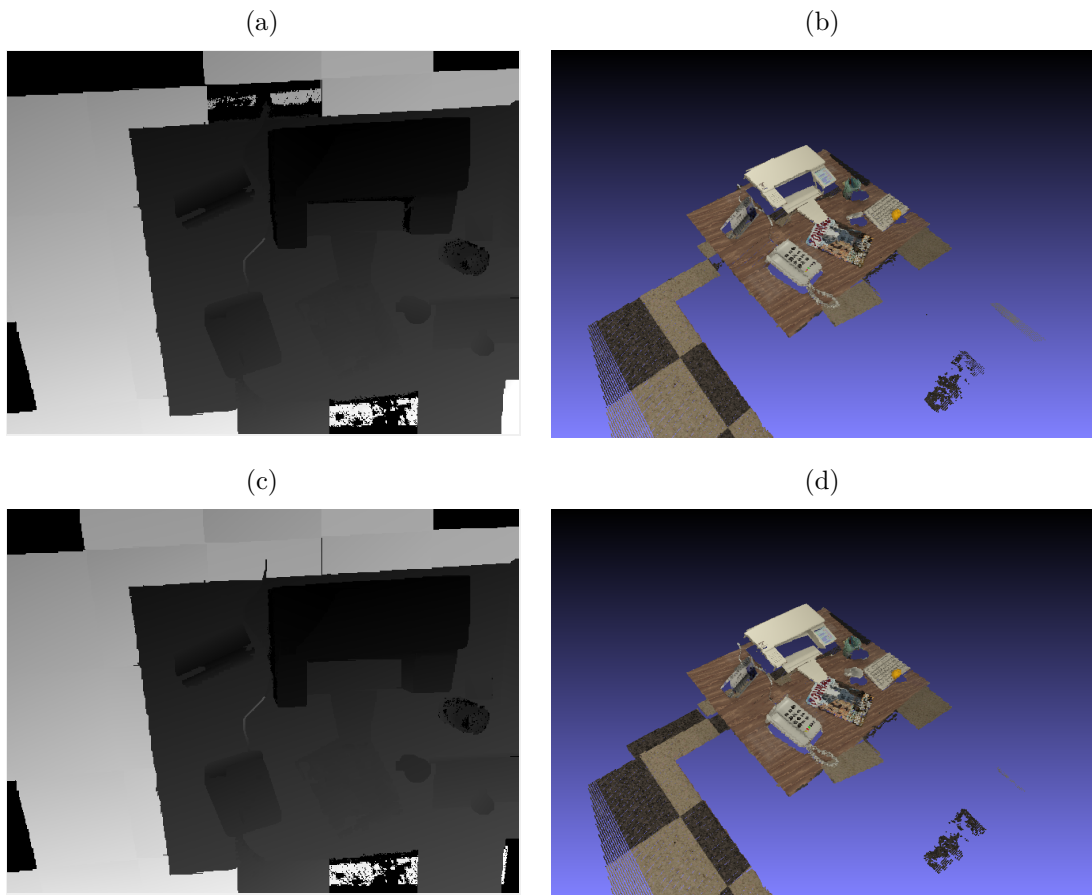
Como comentado na seção anterior, mesmo que o resultado do SAD considerado como melhor (SAD2) tenha apresentado um *recall* inferior ao do experimento SAD4 com verificação LRC sem a utilização de pirâmides, ambos atingiram valores de *recall* semelhantes após a etapa de densificação, tanto para a aproximação por planos, quanto por Delaunay.

Pode-se ainda notar que os erros absolutos máximos continuaram na faixa de 800 *cm* para vários resultados. Isso acontece pelo fato das duas abordagens de densificação da nuvem de pontos não descartarem os *outliers* que não seriam usados para estimação densa. No caso da aproximação por planos, são as áreas cujo plano resultante não contém *inliers* correspondentes a pelo menos 65 % dos pixels totais reconstruídos. Já para o caso da aproximação por Delaunay, são pontos dados como *outliers* e não utilizados para a formação dos triângulos. Entretanto, apesar de não utilizados para a densificação, não são excluídos da nuvem.

Em resumo, os resultados de cada métrica que apresentaram o melhor compromisso

entre precisão e velocidade, considerando as discussões da seção anterior, foram os que verificaram o LRC e utilizaram pirâmide gaussiana novamente (SAD2 e FNCC2, Tabelas 2 e 3). O mapa de profundidade e a nuvem de pontos dos melhores resultados de cada métrica da etapa de reconstrução 3D densa, para ambas abordagens de aproximação, podem ser conferidos nas Figuras 81 e 82.

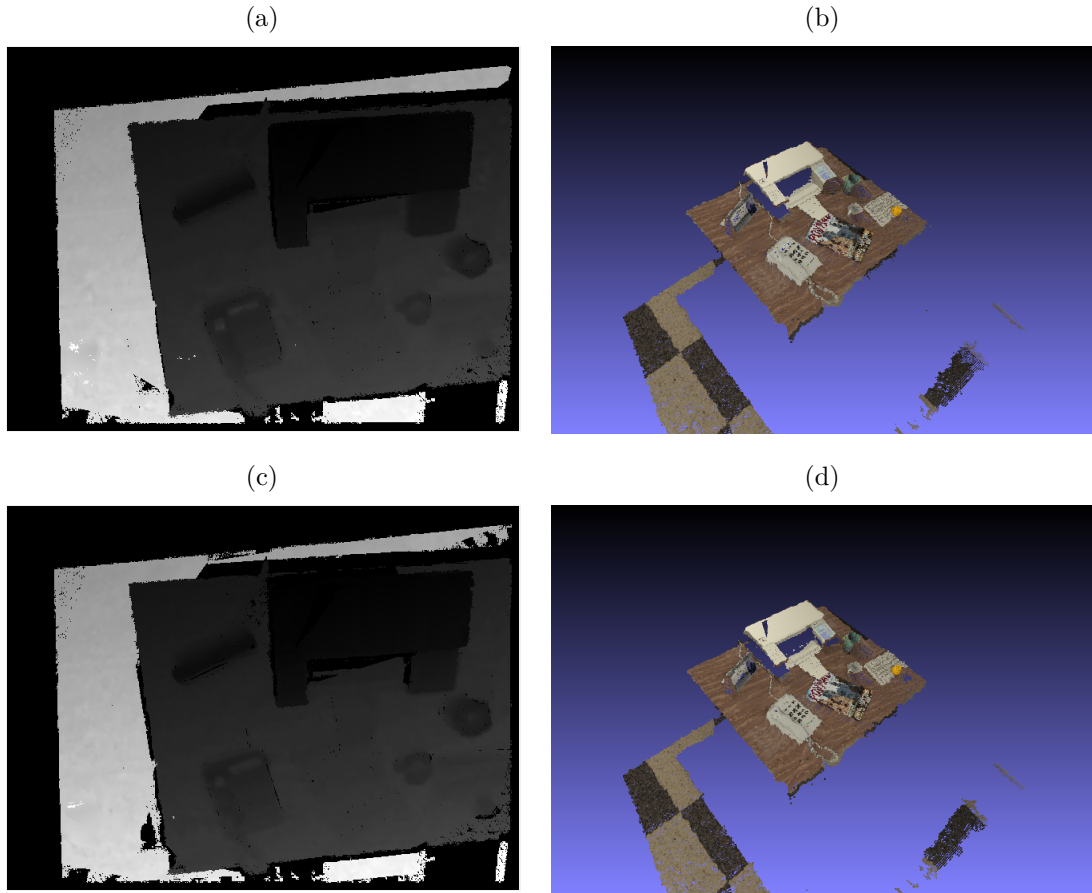
Figura 81 – Mapas de profundidade dos resultados das métricas SAD2 (a) e FNCC2 (c) para a etapa de reconstrução 3D densa por aproximação de planos e suas respectivas nuvens de pontos (b,d).



Fonte: Produção do próprio autor.

Como pode-se notar nos mapas de profundidade e na nuvem de pontos apresentados nas Figuras 81 e 82, os dois resultados da aproximação por planos obtiveram um erro absoluto maior que os da aproximação por Delaunay. Isso aconteceu pois duas regiões do chão acabaram tendo a profundidade estimada como um plano na altura da mesa. Fator esse resultante tanto da segmentação equivocada de uma das áreas quanto da falta de pixels bem reconstruídos, na outra região segmentada, que pudessem gerar um plano correto. Tal separação conseguiu ser feita pela aproximação por Delaunay. A aproximação de regiões por planos é sensível à segmentação incorreta, entretanto, essa sensibilidade pode ser amenizada ao se restringir essa aproximação planar para triângulos formados a

Figura 82 – Mapas de profundidade dos resultados das métricas SAD2 (a) e FNCC2 (c) para a etapa de reconstrução 3D densa por aproximação de superfície por Delaunay e suas respectivas nuvens de pontos (b,d).



Fonte: Produção do próprio autor.

cada três pontos pela triangulação de Delaunay.

Além disso, é possível notar que a aproximação por Delaunay obteve um resultado visualmente melhor que os resultados por aproximação de planos. Observa-se também que uma pequena região da tampa da impressora não foi reconstruída nos resultados da aproximação por Delaunay. Este fato se justifica pela não existência de pontos na divisória das duas regiões segmentadas pelo SRM na tampa da impressora, por esses pontos terem sido descartados durante a etapa de reconstrução esparsa, o que implica na impossibilidade de geração de triângulo para esses pontos.

Finalmente, dos quatro melhores resultados, os que se destacaram nas duas abordagens foram os da métrica SAD (SAD2, Figuras 81b e 82b), por apresentarem valores de erro médio e *recall* próximos aos do FNCC e por serem computacionalmente menos custosas.

Tabela 4 – Dois resultados de reconstrução 3D densa que se destacaram na Seção 4.2 e resultados da técnica do DTAM.

SAD2								
Técnica de densificação	Técnica*		Erro (cm)					Recall (%)
	LRC	Pyr	Média	Mediana	Desvio Padrão	Mínimo**	Máximo	
Planos	✓	✓	5,05	0,67	19,10	0	810,52	90,9
Delaunay	✓	✓	2,28	0,66	10,43	0	810,52	71,0

DTAM							
Etapa	Erro (cm)					Recall (%)	
	Média	Mediana	Desvio Padrão	Mínimo**	Máximo		
Rec. inicial	24,27	8,28	30,98	0	168,17	100	
Rec. final	23,09	4,06	45,39	0	222,37	100	

* Técnicas utilizadas no algoritmo: consistência esquerda-direita (LRC) e pirâmide gaussiana de 2 níveis (Pyr).

** Valores nulos de erro são, na verdade, valores menores que 10^{-5} .

4.3 Comparação dos resultados com o DTAM

Nessa seção, os dois resultados de reconstrução 3D densa que se destacaram na Seção 4.2 (SAD2 para aproximação por planos e por triangulação de Delaunay) serão comparados com os resultados do DTAM da sua estimativa inicial (Rec. inicial) e após o processo de minimização (Rec. final). Na seção seguinte, um resultado de reconstrução 3D densa mais completa da sala onde se encontra a mesa será apresentado.

Os dois resultados de reconstrução 3D densa que se destacaram na Seção 4.2 (Figuras 81b e 82b) e os resultados da técnica do DTAM estão apresentados na Tabela 4.

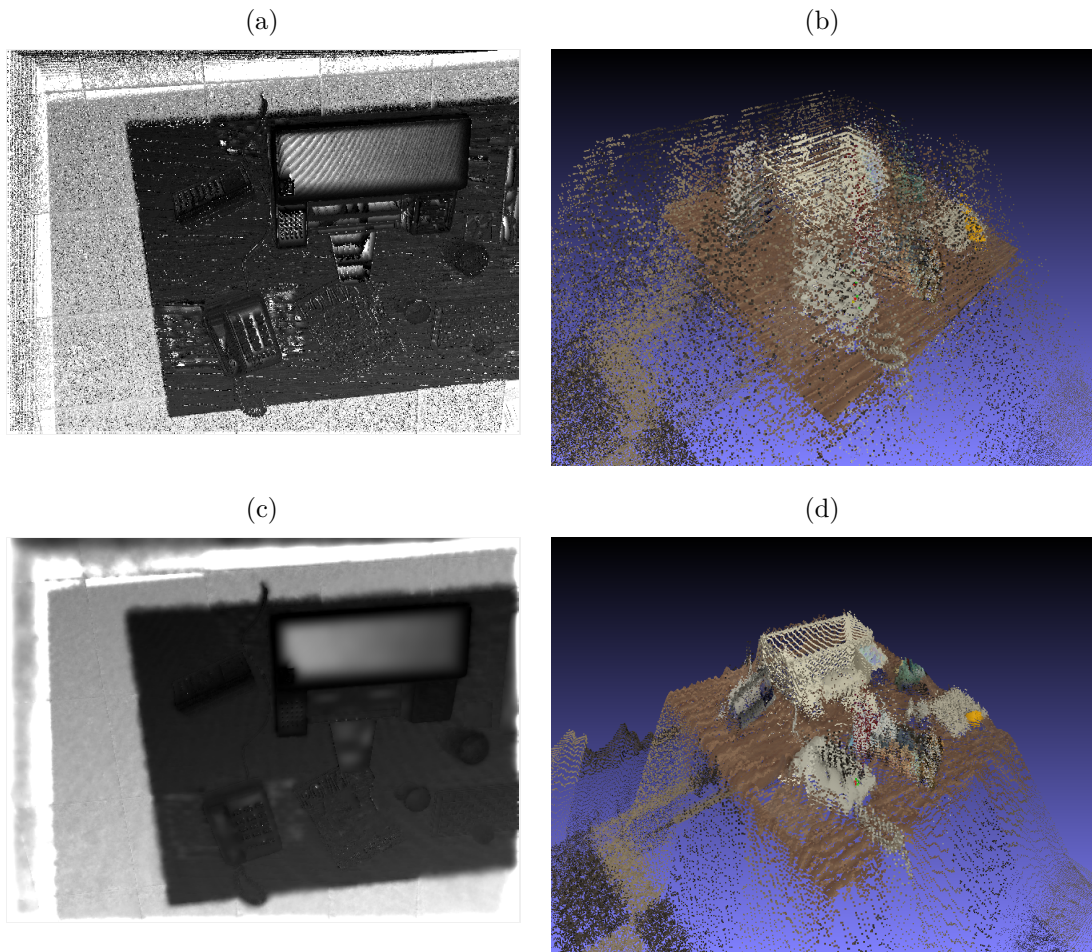
O mapa de profundidade e a nuvem de pontos dos resultados das duas etapas do DTAM podem ser conferidos na Figura 83.

Verifica-se na Tabela 4 que o método do DTAM estima a profundidade de todos os pixels da imagem (*recall* de 100 %) mas, em média, o erro absoluto da sua reconstrução final é 7,3 vezes maior que os obtidos pelo algoritmo proposto, e sua mediana 6,1 vezes.

Nota-se que mesmo para um *recall* próximo ao do DTAM (caso de aproximação por planos), o método proposto obtém resultados melhores que os apresentados pelo DTAM. Um dos motivos é que mesmo reconstruindo mais pixels, a precisão do DTAM acaba sendo muito prejudicada pela profundidade estimada em áreas homogêneas.

É possível observar na Figura 83c erros altos de estimativa no mapa de profundidade sobre diversas áreas da impressora e próximo ao telefone (áreas claras que deveriam ser escuras, vide Figura 77). Por não possuírem textura, essas regiões não são bem reconstruídas,

Figura 83 – Mapas de profundidade da Rec. inicial (a) e Rec. final (c) do DTAM para 30 imagens e $N = 32$ níveis de profundidade, e suas respectivas nuvens de pontos (b,d).



Fonte: Produção do próprio autor.

e essa estimativa, quando muito grosseira, acaba sendo propagada para a reconstrução final (de Rec. inicial para Rec. final). No método aqui proposto, os suspixels dessas regiões homogêneas são excluídos, o que evita que situações como essa ocorram. Vale lembrar que o DTAM não utiliza métricas baseadas em janelas e busca reconstruir todos os pixels da imagem sem nenhuma inferência a mais da cena (como a aproximação de regiões homogêneas por planos ou superfícies) pelo seu objetivo ser abaixar o tempo de processamento, o que muitas vezes prejudica a precisão de seus resultados (Figura 77).

Em suma, os resultados do método proposto obtiveram uma precisão melhor que o DTAM, principalmente em regiões homogêneas. Além disso, apesar dos resultados apresentados pelo algoritmo proposto não atingirem um tempo compatível com aplicações de tempo real, como o DTAM consegue, eles apresentaram um bom compromisso entre qualidade de reconstrução e velocidade de execução. Vale comentar que Newcombe, Lovegrove e Davison (2011) utilizam um sistema formado por um computador *i7 quad-core*

equipado com uma placa de vídeo NVIDIA GTX 480 para cálculos em GPU. No caso do trabalho proposto, além de não se dispor de tal equipamento, o código não foi otimizado. Desta forma, é possível se obter uma velocidade maior de reconstrução com o método proposto, sem que se perca em precisão.

4.4 Ampliando reconstruções prévias

Para se apresentar um resultado mais completo da sala em que a mesa da imagem de referência dos resultados apresentados nesse capítulo está situada, o método proposto foi executado para outras imagens do banco de dados disponível em (HANDA et al., 2012). Desse novo conjunto de 11 imagens, a imagem de referência está apresentada na Figura 84 juntamente da sua segmentação por SRM.

Figura 84 – Imagem de referência do novo conjunto de 11 imagens (a) e sua segmentação por SRM (b). Conjunto esse que apresenta um outro ângulo da sala cena apresentada no conjunto de imagens utilizado para a obtenção dos resultados do Capítulo 4.

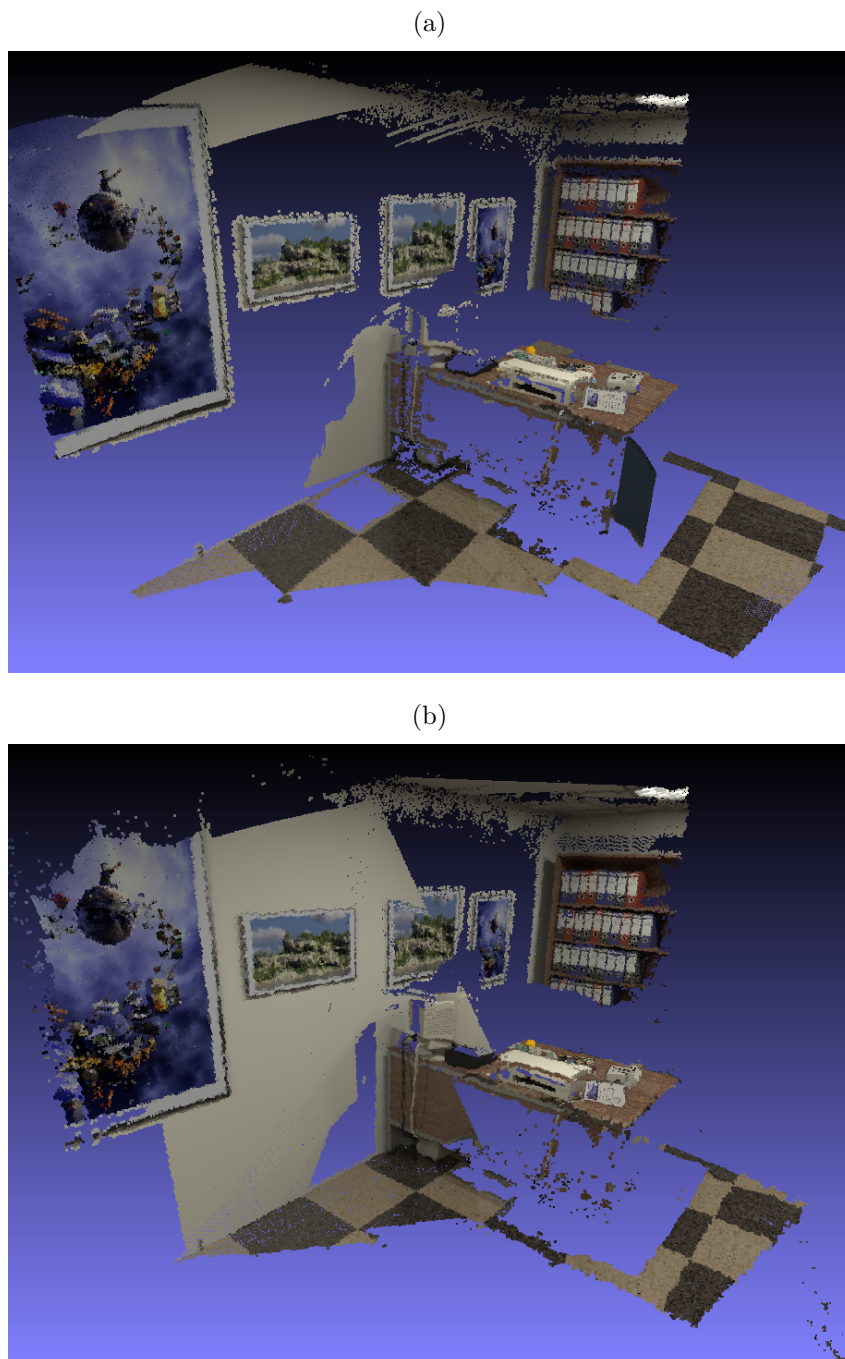


Fonte: Produção do próprio autor.

Na Figura 85 pode-se conferir a junção da nuvem de pontos dos dois resultados de reconstrução 3D densa que se destacaram na Seção 4.2 (SAD2, Figuras 81b e 82b) com os resultados de seus correspondentes do novo conjunto de imagens, considerando que foram utilizadas as mesmas métricas, técnicas e abordagens de aproximação que os dois resultados anteriores. Para os resultados da Figura 85 não é realizada nenhuma operação de fusão de nuvem de pontos. As nuvens de pontos são somente apresentadas em conjunto, utilizando o referencial da primeira nuvem de pontos como global.

Pode-se notar na Figura 85 que a aproximação por planos é de fato sensível a segmentação incorreta de planos (regiões com planos muito incorretos: área lateral da mesa e área da parede segmentada em cor verde (Figura 84b)). Essa sensibilidade é amenizada

Figura 85 – Nuvens de pontos apresentadas em conjunto dos resultados de SAD com verificação de LRC e utilizando pirâmide gaussiana para as imagens de referência das Figuras 76 e 84. Em (a) foi utilizada a densificação da nuvem por aproximação por planos, em (b) por Delaunay.



Fonte: Produção do próprio autor.

ao se restringir essa aproximação planar para triângulos formados a cada três pontos pela triangulação de Delaunay.

Além disso, é possível observar que a Figura 84 possui uma região homogênea

muito grande (região da parede), onde muitos de seus pixels foram reconstruídos para o caso da aproximação por Delaunay (Figura 85b). Já para o caso de aproximação por planos (Figura 85a), isso não aconteceu, pois os pontos reconstruídos na maior região segmentada da parede não conseguiram gerar um plano que contemplasse pelo menos 65 % dos pontos reconstruídos como *inliers*.

5 Conclusões e trabalhos futuros

Neste trabalho foram propostas duas metodologias de reconstrução 3D densa que representam um bom compromisso entre precisão e velocidade, aproximando-se regiões homogêneas por planos e por uma superfície obtida a partir da triangulação de Delaunay. Nos dois casos a imagem de referência é segmentada em regiões homogêneas de cor ou textura similar utilizando a técnica SRM.

O método proposto foi testado para as métricas SAD, SSD e FNCC e comparado com o método DTAM, o qual representa um bom compromisso entre tempo de processamento e qualidade de reconstrução, sendo utilizado em aplicações de tempo real. Para cada métrica, analisou-se a utilização de pirâmide gaussiana para diminuir o tempo de processamento, e a verificação da consistência esquerda-direita.

Dentre as três métricas, o SAD apresentou os menores valores de tempo de processamento em cada experimento. A verificação da consistência esquerda-direita reduziu o erro absoluto da reconstrução, mesmo que diminuindo um pouco o *recall*. Entretanto, essa verificação quase que dobra o tempo de execução do algoritmo. Para abaixar esse tempo, utilizou-se a pirâmide gaussiana de multiresolução.

Essa combinação possibilitou uma reconstrução 3D densa com um *recall* de 90,9 % para a abordagem de aproximação por planos e 71,0 % para a abordagem de aproximação por Delaunay. Mesmo sem utilizar um método de otimização, a técnica proposta se destaca por atingir o *recall* de 71,0 % com uma precisão dez vezes maior que a conseguida pelo DTAM. Vale ressaltar que devido à sobreposição máxima entre as imagens corresponderem a 82,5 % da imagem de referência, esse acaba sendo o *recall* máximo possível para a densificação da nuvem de pontos pela aproximação por Delaunay. Ou seja, um *recall* de 71,0 % corresponde a 86,1 % do *recall* máximo, sendo assim um bom resultado. Tal resultado pode ser considerado promissor e sugere que a metodologia proposta pode ser utilizada para aplicações futuras do grupo de pesquisa do laboratório.

Em suma, os resultados do método proposto obtiveram uma precisão melhor que o DTAM, principalmente em regiões homogêneas. Apesar dos resultados apresentados pelo algoritmo proposto não atingirem um tempo compatível com aplicações de tempo real, como o DTAM consegue, eles apresentaram um bom compromisso entre qualidade de reconstrução e velocidade de execução. Vale comentar que Newcombe, Lovegrove e Davison (2011) utilizam um sistema formado por um computador *i7 quad-core* equipado com uma placa de vídeo NVIDIA GTX 480 para cálculos em GPU. Já os experimentos realizados para o trabalho proposto dispuseram de um *hardware* com menos capacidade de processamento. Além disso, o código implementado não está otimizado. Desta forma, é

possível se obter velocidades maiores de reconstrução com o método proposto, sem que se perca em precisão.

De uma maneira geral, o trabalho alcançou seus objetivos. Mesmo assim, seria interessante em um trabalho futuro que o método proposto também fosse implementado para um sistema monocular de movimentos predominantemente em linha reta. Assim, seria possível abranger os casos onde a câmera se movimenta para frente, filma o caminho percorrido e realiza ao mesmo tempo a reconstrução 3D da cena. Além disso, seria pertinente estender o método proposto para os casos em que se utiliza sistemas de câmeras estéreo, de forma que possibilitaria, por exemplo, um robô realizar a reconstrução 3D mesmo quando parado.

Outra limitação do sistema está no processo de pirâmide multiresolução, onde todos os suspixels de um nível superior propagam para o próximo nível como quatro pixels que serão descartados. Isso reduz o *recall* ainda mais para cada nível de pirâmide adicionado. Talvez fosse interessante analisar uma forma de não se descartar esses pixels, e sim estender a procura por seu correspondente na linha epipolar de z_{min} a z_{max} no próximo nível ou, até mesmo, estender o resultado dos vizinhos para esses pontos.

Mesmo não descartando os pixels, a pirâmide tem o objetivo de diminuir a complexidade da etapa de *matching* e acelerar o processo de reconstrução. Para esse fim, neste trabalho, foi analisada a utilização de pirâmides gaussianas. Entretanto, trabalhos como (MALATHI; BHUYAN, 2015; PRABHAKAR; JYOTHI, 2012) utilizam Wavelets nessa etapa de correspondência por já apresentarem essa característica de pirâmide. Logo, poderia-se analisar as vantagens e desvantagens desses dois métodos a fim de concluir quando e qual deles utilizar.

Por fim, é importante relembrar que uma técnica de segmentação de imagem em regiões homogêneas não tem a finalidade de encontrar áreas planares mas sim regiões similares. Dessa forma, nada impede que outros métodos possam ser usados para segmentar as imagens, dependendo das características e aspectos das cenas. Sendo assim, outras técnicas como o JSEG (DENG; MANJUNATH, 2000) também podem ser testadas em trabalhos futuros.

Referências

- ACHANTA, R.; SHAJI, A.; SMITH, K.; LUCCHI, A.; FUA, P.; SUSSTRUNK, S. SLIC Superpixels. *EPFL Technical Report 149300*, n. June, p. 15, 2010.
- AIT-JELLAL, R.; ZELL, A. A fast dense stereo matching algorithm with an application to 3d occupancy mapping using quadrocopters. In: IEEE. *Advanced Robotics (ICAR), 2015 International Conference on*. [S.l.], 2015. p. 587–592.
- ALCANTARILLA, P. F.; BEALL, C.; DELLAERT, F. Large-scale dense 3d reconstruction from stereo imagery. In: *5th Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV13)*. [S.l.: s.n.], 2013.
- AMORIM, L. A.; SANTOS, M. C. P.; QUEIROZ, F. M.; SILVA, L. d. A.; VASSALO, R. F.; FILHO, M. S. Estimação de Posição e Atitude de um Veículo Aéreo Não Tripulado Baseada em GPS, IMU e Dados Visuais. In: *Anais do XII Simpósio Brasileiro de Automação Inteligente (SBAI'2015)*. [S.l.: s.n.], 2015.
- AMORIM, L. A.; SILVA, L. d. A.; QUEIROZ, F. M.; SALVADOR, R. d. M.; VASSALO, R. F.; FILHO, M. S. Construção de Mosaico Utilizando Imagens Aéreas Adquiridas de Forma Autônoma. In: *Anais do XII Simpósio Brasileiro de Automação Inteligente (SBAI'2015)*. [S.l.: s.n.], 2015.
- BAY, H.; TUYTELAARS, T.; GOOL, L. V. Surf: Speeded up robust features. In: SPRINGER. *European conference on computer vision*. [S.l.], 2006. p. 404–417.
- BIGGS, N.; LLOYD, E. K.; WILSON, R. J. *Graph Theory, 1736-1936*. [S.l.]: Oxford University Press, 1976.
- BRADSKI, G. *Dr. Dobb's Journal of Software Tools*, 2000.
- BYLOW, E.; STURM, J.; KERL, C.; KAHL, F.; CREMERS, D. Real-time camera tracking and 3d reconstruction using signed distance functions. In: ROBOTICS: SCIENCE AND SYSTEMS. *Robotics: Science and Systems (RSS) Conference 2013*. [S.l.], 2013. v. 9.
- CHANG, P.-L.; STOYANOV, D.; DAVISON, A. J. et al. Real-time dense stereo reconstruction using convex optimisation with a cost-volume for image-guided robotic surgery. In: SPRINGER. *International Conference on Medical Image Computing and Computer-Assisted Intervention*. [S.l.], 2013. p. 42–49.
- CONCHA, A.; CIVERA, J. DPPTAM: Dense Piecewise Planar Tracking and Mapping from a Monocular Sequence. *Intelligent Robots and Systems (IROS 2015), 2015 IEEE/RSJ International Conference on*, p. 5686–5693, 2015.
- CONCHA, A.; HUSSAIN, W.; MONTANO, L.; CIVERA, J. Manhattan and Piecewise-Planar Constraints for Dense Monocular Mapping. *Roboticsproceedings.Org*, 2014.
- CONCHA, A.; HUSSAIN, W.; MONTANO, L.; CIVERA, J. Incorporating scene priors to dense monocular mapping. *Autonomous Robots*, v. 39, n. 3, p. 279–292, 2015.

- CROW, F. C. Summed-area tables for texture mapping. *ACM SIGGRAPH computer graphics*, ACM, v. 18, n. 3, p. 207–212, 1984.
- DELAUNAY, B. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, v. 7, n. 793-800, p. 1–2, 1934.
- DENG, Y.; MANJUNATH, B. Jseg-segmentation of color-texture regions in images and video. *Vision Research Lab, UCSB*, 2000.
- DIAS, T.; ARAUJO, H.; MIRALDO, P. 3d reconstruction with low resolution, small baseline and high radial distortion stereo images. In: ACM. *Proceedings of the 10th International Conference on Distributed Smart Camera*. [S.l.], 2016. p. 98–103.
- FELZENSZWALB, P. F.; HUTTENLOCHER, D. P. Efficient graph-based image segmentation. *International Journal of Computer Vision*, v. 59, n. 2, p. 167–181, 2004.
- FISCHLER, M. a.; BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, v. 24, n. 6, p. 381–395, 1981. Disponível em: <<http://portal.acm.org/citation.cfm?doid=358669.358692>>.
- FRAUNDORFER, F.; SCHINDLER, K.; BISCHOF, H. Piecewise planar scene reconstruction from sparse correspondences. *Image and Vision Computing*, v. 24, n. 4, p. 395–406, 2006.
- FURUKAWA, Y.; HERNÁNDEZ, C. et al. *Multi-view stereo: A tutorial*. [S.l.]: Citeseer, 2015.
- FURUKAWA, Y.; PONCE, J. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 32, n. 8, p. 1362–1376, 2010.
- FUSIELLO, A.; IRSARA, L. Quasi-euclidean uncalibrated epipolar rectification. In: IEEE. *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. [S.l.], 2008. p. 1–4.
- GEIGER, A.; ROSER, M.; URTASUN, R. Efficient large-scale stereo matching. In: SPRINGER. *Asian conference on computer vision*. [S.l.], 2010. p. 25–38.
- GERIG, G. *3D Computer Vision Spring 2012*. [S.l.]: Utah, School of Computing, 2012.
- GOESELE, M.; CURLESS, B.; SEITZ, S. M. Multi-view stereo revisited. In: IEEE. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. [S.l.], 2006. v. 2, p. 2402–2409.
- HANDA, A.; NEWCOMBE, R. A.; ANGELI, A.; DAVISON, A. J. Real-time camera tracking: When is high frame-rate best? In: *Proc. of the European Conference on Computer Vision(ECCV)*. [S.l.: s.n.], 2012.
- HARTLEY, R.; GUPTA, R. Computing matched-epipolar projections. In: IEEE. *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on*. [S.l.], 1993. p. 549–555.

HARTLEY, R.; ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. Cambridge: Cambridge University Press, 2003.

HARTLEY, R. I. Theory and practice of projective rectification. *International Journal of Computer Vision*, Springer, v. 35, n. 2, p. 115–127, 1999.

HEBERT, M.; ZELLER, C.; FAUGERAS, O.; ROBERT, L. *Applications of non-metric vision to some visually guided robotics tasks*. Tese (Doutorado) — INRIA, 1995.

HIRSCHMULLER, H.; SCHARSTEIN, D. Evaluation of stereo matching costs on images with radiometric differences. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 31, n. 9, p. 1582–1599, 2009.

IZADI, S.; KIM, D.; HILLIGES, O.; MOLYNEAUX, D.; NEWCOMBE, R.; KOHLI, P.; SHOTTON, J.; HODGES, S.; FREEMAN, D.; DAVISON, A.; FITZGIBBON, A. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. New York, NY, USA: ACM, 2011. (UIST '11), p. 559–568. ISBN 978-1-4503-0716-1. Disponível em: <<http://doi.acm.org/10.1145/2047196.2047270>>.

KO, H.; SHIM, H. S.; CHOI, O.; KUO, C.-C. J. Robust uncalibrated stereo rectification with constrained geometric distortions (usr-cgd). *arXiv preprint arXiv:1603.09462*, 2016.

KRUSKAL, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, JSTOR, v. 7, n. 1, p. 48–50, 1956.

LEWIS, J. P. Fast template matching. In: *Vision interface*. [S.l.: s.n.], 1995. v. 95, n. 120123, p. 15–19.

LI, L.; SUN, L.; GUO, J.; LI, S. Image Segmentation Based on Superpixels and Region Merging. v. 23, p. 8787–8797, 2015.

LOWE, D. G. Object recognition from local scale-invariant features. In: IEEE. *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. [S.l.], 1999. v. 2, p. 1150–1157.

MA, Y.; SOATTO, S.; KOSECKA, J.; SASTRY, S. S. *An invitation to 3-d vision: from images to geometric models*. [S.l.]: Springer Science & Business Media, 2012.

MALATHI, T.; BHUYAN, M. Estimation of disparity map of stereo image pairs using spatial domain local gabor wavelet. *IET Computer Vision*, IET, v. 9, n. 4, p. 595–602, 2015.

MALLON, J.; WHELAN, P. F. Projective rectification from the fundamental matrix. *Image and Vision Computing*, Elsevier, v. 23, n. 7, p. 643–650, 2005.

MATAS, J.; CHUM, O.; URBAN, M.; PAJDLA, T. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, Elsevier, v. 22, n. 10, p. 761–767, 2004.

MIKHINA, T. *Tatiana Mikhina*. 2017. <<http://tatianamikhina.com>>. Acessado: 2017-01-30.

- MONASSE, P.; MOREL, J.-M.; TANG, Z. Three-step image rectification. In: BMVA PRESS. *BMVC 2010-British Machine Vision Conference*. [S.l.], 2010. p. 89–1.
- NAVAB, N. *3D Computer Vision II Winter Term 2010/2011*. [S.l.]: Technische Universität München, 2010.
- NEWCOMBE, R. A.; LOVEGROVE, S. J.; DAVISON, A. J. {DTAM}: Dense Tracking and Mapping in Real-Time. *Int. Conf. on Computer Vision (ICCV)*, p. 2320–2327, 2011.
- NOCK, R.; NIELSEN, F. Statistical region merging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 26, n. 11, p. 1452–1458, nov 2004.
- PAGANI, A.; GAVA, C.; CUI, Y.; KROLLA, B.; HENGGEN, J.-M.; STRICKER, D. Dense 3D Point Cloud Generation from Multiple High-resolution Spherical Images. *International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST)*, p. 1–8, 2011.
- PEREIRA, F. G.; VASSALLO, R. F.; SALLES, E. O. T. Human–robot interaction and cooperation through people detection and gesture recognition. *Journal of Control, Automation and Electrical Systems*, Springer, v. 24, n. 3, p. 187–198, 2013.
- PRABHAKAR, C.; JYOTHI, K. A wavelet-based multiresolution approach to stereo matching of face images. In: IEEE. *Advanced Communication Control and Computing Technologies (ICACCCT), 2012 IEEE International Conference on*. [S.l.], 2012. p. 316–320.
- PRADEEP, V.; RHEMANN, C.; IZADI, S.; ZACH, C.; BLEYER, M.; BATHICHE, S. Monofusion: Real-time 3d reconstruction of small scenes with a single web camera. In: IEEE. *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*. [S.l.], 2013. p. 83–88.
- QUEIROZ, F. M.; COVRE, V. B.; RAMPINELLI, M.; PEREIRA, F. G.; VASSALO, R. F.; BASTOS-FILHO, T. F. Localização e Guiagem de um Robô Móvel em um Espaço Inteligente. In: *Anais do XII Simpósio Brasileiro de Automação Inteligente (SBAI'2015)*. [S.l.: s.n.], 2015.
- RAMPINELLI, M.; COVRE, V. B.; QUEIROZ, F. M. de; VASSALLO, R. F.; BASTOS-FILHO, T. F.; MAZO, M. An intelligent space for mobile robot localization using a multi-camera system. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 14, n. 8, p. 15039–15064, 2014.
- RAMPINELLI, M.; QUEIROZ, F. M.; COVRE, V. B.; VASSALO, R. F.; BASTOS-FILHO, T. F. Calibração Automática de um Sistema Multi-Câmeras Utilizando Robô Móvel. In: *Anais do XII Simpósio Brasileiro de Automação Inteligente (SBAI'2015)*. [S.l.: s.n.], 2015.
- RIEDL, W. F.; SEFIDGAR, R. *Kruskal's Algorithm*. 2014. <<https://www-m9.ma.tum.de/graph-algorithms/mst-kruskal>>. Technische Universität München. Acessado: 2017-03-01.
- ROMANONI, A.; MATTEUCCI, M. Incremental reconstruction of urban environments by edge-points delaunay triangulation. In: IEEE. *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. [S.l.], 2015. p. 4473–4479.
- SA, F. B.; CYPRIANO, M. F.; PEREIRA, F. G.; VASSALO, R. F.; NETO, A. F. Planejamento de Trajetória para um Robô Móvel usando Imagens capturadas por um VANT. In: *Anais do XX Congresso Brasileiro de Automação (CBA'2014)*. [S.l.: s.n.], 2014.

- SANDERSON, C.; CURTIN, R. Armadillo: a template-based c++ library for linear algebra. *JOSS*, The Open Journal, v. 1, n. 2, jun 2016.
- SCOPUS. 2016. <<http://www.scopus.com>>. Elsevier. Acessado: 2016-10-29.
- SILVA, L. d. A.; LUBE, J. G. P.; VASSALLO, R. F. Aproximação Planar por Partes para Reconstrução 3D Densa. In: *Anais do XXI Congresso Brasileiro de Automática (CBA'2016)*. [S.l.: s.n.], 2016.
- STRECHA, C.; TUYTELAARS, T.; GOOL, L. V. Dense matching of multiple wide-baseline views. In: IEEE. *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. [S.l.], 2003. p. 1194–1201.
- TANSKANEN, P.; KOLEV, K.; MEIER, L.; CAMPOSECO, F.; SAURER, O.; POLLEFEYS, M. Live metric 3d reconstruction on mobile phones. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2013. p. 65–72.
- WU, T.-P.; YEUNG, S.-K.; JIA, J.; TANG, C.-K. Quasi-dense 3D reconstruction using tensor-based multiview stereo. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.]: IEEE, 2010. v. 4, p. 1482–1489.